NPS62-79-018PR

NAVAL POSTGRADUATE SCHOOL

Monterey, California

SOFTWARE DEVELOPMENT FOR

A SATELLITE SIGNAL ANALYZER

Todd Bruner
John E. Ohlson

December 1979

Project Report

Prepared for: Naval Electronic Systems Command
PME-106-1
Washington, D.C. 20360

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral T.F. Dedman
Superintendent

Jack R. Borsting
Provost

The work reported herein was supported in part by the Naval Electronic Systems Command, PME-106-1.

This report was prepared by:

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NPS62-79-018PR	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): SOFTWARE DEVELOPMENT FOR A SATELLITE SIGNAL ANALYZER	5. TYPE OF REPORT & PERIOD COVERED: Project Report
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Todd Thornton Whitney Bruner John E. Ohlson	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: N0003980WR09137
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Electronic Systems Command Washington, D.C. 20360	12. REPORT DATE: December 1979
	13. NUMBER OF PAGES: 199
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Satellite Signal Analyzer
Software Development

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A Satellite Signal Analyzer is being constructed by the Satellite Communications Laboratory of the Naval Postgraduate School. The purpose of this system is to provide high-speed spectrum analysis and characterization of the outputs of UHF communication satellite transponders while in orbit and operating. It is constructed around an INTERDATA 7/32 minicomputer which provides all the necessary control for most of the equipment in the system. Fast Fourier Transforms (FFT's) are provided by the AP-12ØB Array

Processor manufactured by FLOATING POINT SYSTEMS. Extremely accurate frequency measurement is provided by phase-locked loop receivers. A HEWLETT-PACKARD 5061A Cesium Beam Frequency Standard provides reference frequencies for all measurement and synthesized equipment.

This thesis documents a portion of the software development; specifically presented are documentation for (a) the NPS SATCOM Analyzer System; (b) the Spectrum Analyzer Control software; and (c) diagnostic procedures.

ABSTRACT

A Satellite Signal Analyzer is being constructed by the Satellite Communications Laboratory of the Naval Postgraduate School. The purpose of this system is to provide high-speed spectrum analysis and characterization of the outputs of UHF communication satellite transponders while in orbit and operating. It is constructed around an INTERDATA 7/32 minicomputer which provides all the necessary control for most of the equipment in the system. Fast Fourier Transforms (FFT's) are provided by the AP-12ØB Array Processor manufactured by FLOATING POINT SYSTEMS. Extremely accurate frequency measurement is provided by phase-locked loop receivers. A HEWLETT-PACKARD 5Ø61A Cesium Beam Frequency Standard provides reference frequencies for all measurement and synthesized equipment.

This thesis documents a portion of the software development; specifically presented are documentation for (a) the NPS SATCOM Analyzer System; (b) the Spectrum Analyzer Control software; and (c) diagnostic procedures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This page intentionally blank.

I. INTRODUCTION

A. BACKGROUND

This project, which has been funded by PME 1Ø6-1 of the Naval Electronic Systems Command (NAVELEX), is part of a series of Radio Frequency Interference (RFI) measurements and analysis undertaken by the Satellite Communications (SATCOM) Laboratory of the Naval Postgraduate School (NAVPGSCOL). Previous efforts include evaluation of the AS3Ø18/-WSC-1(v) shipboard SATCOM antenna [1], preparation of a shipboard RFI measurement package [2-5], evaluation of shipboard RFI [6], construction of a shipboard RFI simulator [7], and measurement of shipboard SATCOM terminal performance in the presence of specific RFI sources [8-1Ø].

In March of 1977, this laboratory received funding from PME 1Ø6-1 to develop, design and construct a SATCOM Signal Analyzer at NAVPGSCOL. The purpose of this system is to provide high-speed spectrum analysis and characterization of the outputs of ultrahigh frequency (UHF) communication satellite transponders while operating in orbit. This system is to develop techniques for use in the design of a SATCOM monitoring system for use in Naval Communication Stations. The first reports on this effort include receiver design for the satellite signal analyzer [11], digital control and processing for a satellite communications monitoring system [12], and hardware development for the satellite signal analyzer [13].

B. SPECIFIC GOALS

The specific goals in the development of this system is (1) to develop the necessary equipment to make real-time signal measurements at the Naval Postgraduate School, and (2) to provide the necessary research and development of real-time signal analysis techniques and equipment for use in a follow-on version of the Fleet Satellite Communications Spectrum Monitor (FSM) presently in use at Naval Communications Stations to monitor the operations of the Fleet Satellite (FLTSAT) and GAPFILLER satellites.

C. SCOPE OF THIS PROJECT

This report documents the software development of spectrum analysis control as an applications program. Beyond the documentation, an important aspect of this report is a description of the various procedures used to work the SATCOM Signal Analyzer (SSA).

II. SATCOM SIGNAL ANALYZER

A. COMPONENTS

The SSA is constructed around an Interdata 7/32 minicomputer which provides control for most of the equipment in the system. Control of some counters and frequency synthesizers as an interim measure is achieved by the Hewlett-Packard 983ØA programmable calculator. A few control functions are done manually. There are three primary lines used in the SSA for the purpose of control and digital data. They are the IEEE standard digital interface bus (IEEE 488), for communication with all Hewlett-Packard and Systron Donner equipment; the RS-232C standard data communication link, for communication with most I/O devices; and the Interdata modified interface bus [12]. The reference frequencies to all measurement and synthesized equipment is provided by a Hewlett-Packard 5Ø61A cesium beam frequency standard. The probe, main, and spectrum receivers are the first points where the analog signal come under computer control. At these points digital information is obtainable for all receivers except the spectrum receiver. Although the spectrum receiver is under computer control, the analog signal is digitized only in the data acquisition unit (DAU). Once the analog information has been converted to some form of digital information then mathematical algorithms further manipulate the information in the Floating Point Systems AP-12ØB

array processor. The Fast Fourier Transform (FFT) is just one of many algorithms that are implemented in the AP-12ØB, but certainly the FFT is the one of primary note. The last components of the SSA perform the function of I/O. In most cases the Tektronix 4Ø14 terminal is providing information about spectrum or probe analysis, whereas the printer, disc, and tapes are devices for mass storage, utilities, and program development.

B. SIGNAL FLOW AND CONVERSION

Figure 2-1 shows the SSA signal flow and conversion. A1, the RF Group, has a number of functions. The RF Group serves as the SSA front end. The RF Group distributes the received signal to the AN/WSC-3 receiver, the A2 blanker and downconverter, and the HP 8554B spectrum analyzer. The RF Group transmits signals. Transmission is from the AN/WSC-3 transmitter and the A2Ø Probe Transmitter. The RF Group provides satellite transponder simulation. These functions are not under computer control. All signals are analog.

The second channel in the signal flow is the A2 Blanker/ Downconverter. This unit provides frequency translation of the received RF to a first intermediate frequency band of 6Ø-9Ø MHz. It also delays the received signal until the blanker channel can perform the blanking logic. Control of this unit is manual. The signal is still in analog form.

A1 and A2 do not have computer control, nor do they need

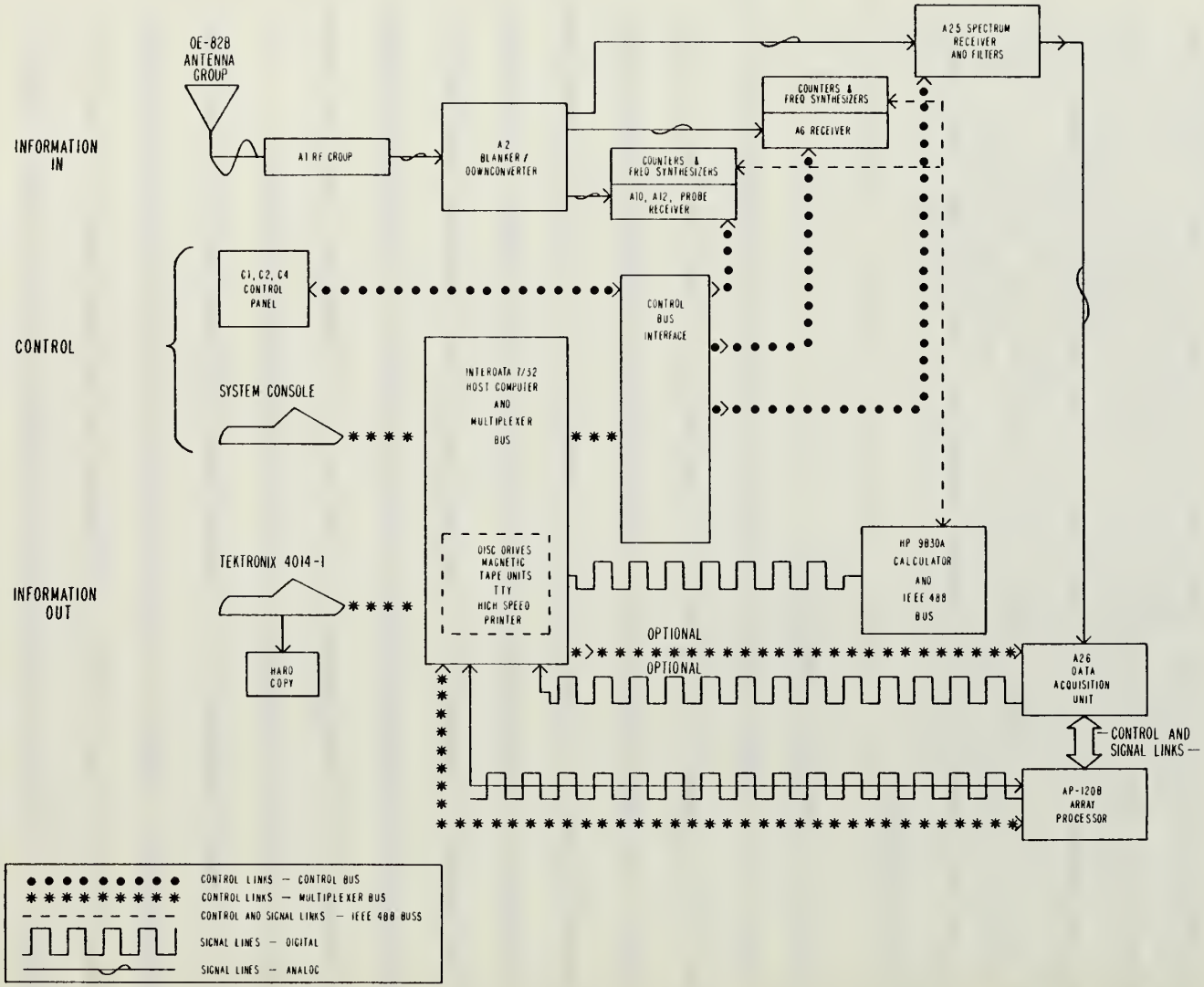


Figure 2-1

SSA SIGNAL FLOW AND CONVERSION

it. Their operations are discussed in [13]. It is at the receivers that the first elements of the SSA come under computer control. The probe receivers (A1Ø and A12) and the main receiver A6 operations are discussed in [11 and 14]. These receivers are under control via the control bus vice RS-232 or IEEE 488. The local oscillators to the receivers are under computer control via either the IEEE 488 and HP 983ØA or the control bus depending on the frequency synthesizer. For A1Ø, A12, and A6 the primary information, frequency, is derived by electronic counters. It is at this point that analog information is converted to digital information. The counters are indicating frequency. The HP 983ØA has the ability to read the counters and the Systron Donner clock. This information can be stored on tape and manipulated either by the Interdata 7/32 and AP-12ØB or by the HP 983ØA alone.

The spectrum receiver is different. It is the first point of computer control for the analog signal that is to be used to derive spectrum. But there is no conversion to digital information in this receiver. Instead there exists the ability to choose different receiver channels. These channels, so as not to confuse them with the satellite channels, will be referred to as Spectrum Receiver Channels (SPCHNL A, B or C). There are also variable filters that may be chosen for spectrum receiver channel C. The filter bandwidths of these filters are 3kHz, 1ØkHz, 3ØkHz, 1ØØkHz,

and 1MHz. The third parameter to the receiver that may be changed is the attenuation. Ø-127dB of attenuation can be controlled by the computer, and Ø-59dB can be set by manual means. The analog signal basically is modified by this receiver by down conversion to 3ØMHz, by filtering, and by attenuation. The A26 data acquisition unit (DAU) modifies the signal further by down conversion to base band and then doing analog to digital conversion. It is here that the analog signal is converted to a digital form.

Digital information is available to the host computer for processing from the HP 983ØA or from the DAU. (There is a software oriented problem associated with the HP 983ØA. The software driver is the original driver written for the HP 983ØA for the purpose of demonstration and feasibility. It has not been changed since. The problem experienced is that when taking data from the HP 983ØA the Interdata 7/32 and HP 983ØA get out of synchronization. When operating under one operating system only about one hour of communication is observed before synchronization is lost. With the same driver in a different operating system the synchronization is lost after about a minute. Two IEEE Instrumentation Bus Controllers have been delivered and will replace the HP 983ØA as a controller. It is expected that when the driver for these controllers is written that the present problem associated with HP 983ØA will be solved.) The digital information is then processed by an array processor (Floating

Point Systems AP-12ØB). The results of this processing are then available for storage or display. Storage can be done on cassette tapes, 2.5 M byte disc, and 9-track magnetic tape. Display can be observed in graphic form on the Tektronix 4Ø14 terminal. The Tektronix terminal has enhanced graphic capabilities and can operate at 154 K baud. A hard copy unit is connected to the Tektronix terminal. Other display devices include a high speed printer, teletype, and keyboard/CRT system.

Increasing throughput by reducing the time that the host computer uses to provide communication between the source of the digital information and the point of processing is desirable. This can be achieved by directly interfacing the DAU to the AP-12ØB. The nature of the information from the probe and main receiver does not require this interfacing. The option of directly interfacing the DAU to the AP-12ØB requires some hardware modifications to the DAU [15] and a software package for the AP-12ØB. The hardware modifications are complete. The software development is still in progress.

C. METHOD OF COMPUTATION (SPECTRUM)

For cohesiveness, the method of computation and capability some of which was presented in [12, 18], are now presented. Dual channel, inphase and quadrature, down conversion of the signal is used to produce two baseband sig-

nals which are sampled simultaneously, held and analog-to-digital converted to form the "real" and "imaginary" components of a complex waveform. The bandpass characteristics of the RF are all preserved as demonstrated below and illustrated in Figure 2-2.

Local Oscillator Signal: $\cos(\omega_L t)$
RF (or IF) Signal: $X(t)$
"In-Phase" Component: $I(t)=\cos(\omega_L t)x(t)$
"Quadrature Phase: Component: $Q(t)=\sin(\omega_L t)x(t)$

Combining as "real" and "imaginary" components of a "complex" signal:

$$\begin{aligned} y(t) &= I(t) + jQ(t) \\ &= \cos(\omega_L t)x(t) + j\sin(\omega_L t)x(t) \\ &= \overline{/}\cos(\omega_L t) + j\sin(\omega_L t)\overline{/}x(t) \\ &= \exp(j\omega_L t)x(t) \end{aligned}$$

and taking the Fourier transform,

$$\begin{aligned} F\overline{/}y(t)\overline{/} &= F\overline{/}\exp(j\omega_L t)x(t)\overline{/} \\ &= F_x(\omega-\omega_L) \end{aligned} \qquad (1)$$

Thus, the D.C. component in the baseband is the component in the RF signal at the frequency of the local oscialltor. Continuing from Equation 1:

$$F\overline{/}y(t)\overline{/} = y(f) = \int_{-\infty}^{\infty} x(t)\exp(j2\pi f_L t)\exp(j2\pi ft)dt$$

$$= \int_{-\infty}^{\infty} x(t)\exp\ (-j2\pi(f-f_L)$$

Restated as the discrete Fourier transform (DFT):

$$y_d(K\Delta f) = \sum_{n=\emptyset}^{n=N=1} x(n\tau)\exp(-j2\pi K\Delta fn\tau)$$

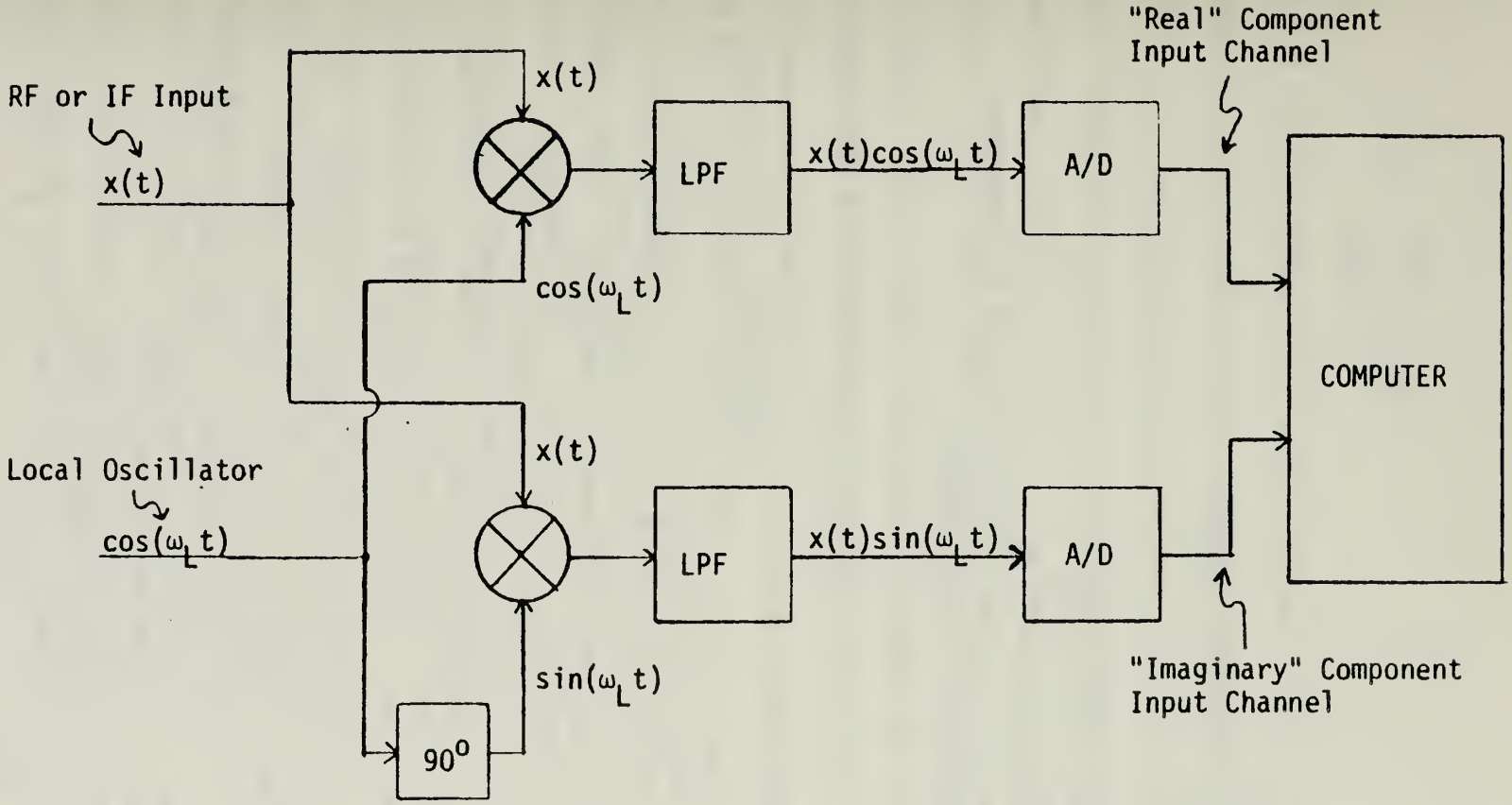


Figure 2-2

Dual Channel Down-Conversion Technique.

letting

$$\Delta f = \frac{2}{N\tau}$$

$$y_d(k\Delta f) = \sum_{n=\emptyset}^{n=N-1} x(n\tau)\exp(-j2\pi\frac{Kn}{N}) \qquad (2)$$

Where N = number of samples

τ = sampling interval

Δf = spacing of the equally spaced k discrete elements of the spectra

K = $\emptyset$, 1, 2, ---N-1

$y_d(k\Delta f)$ is the frequency spectrum composed of K discrete elements having magnitudes able to be calculated from equation (2). $x(n\tau)$ is the signal, a time domain function, from which data have been acquired in the form of N sampled amplitudes, the samples being taken once every τ seconds. The sampling rate or sampling frequency is $1/\tau$ samples/second.

The DFT still presents problems in that it requires lengthy calculations. The calculations can be reduced by use of the Fast Fourier Transform (FFT) algorithm. Many of the terms created by substituting integral values of n and k into equation (2) are identical due to the $\exp(-j2\pi kn/N)$ expression which is inherently periodic.

Continuing by rewriting equation (2):

$$y(k) = \sum_{n=\emptyset}^{N-1} x(n)\exp(-j\frac{2\pi}{N}nk)$$

replacing $\exp(-j\frac{2\pi}{N})$ by W we have

$$y(k) = \sum_{n=\emptyset}^{N-1} x(n)W^{nk}$$

As can be seen that for each sampled data value input, a summation informed of the product between the original data sample and each sine/cosine values represented by one complete circulation of the rotating parameter W^{nk}. W can be considered a point of the unit circle, and W^N is also a point on the circle with an angle n times the angle of W. The above general FFT algorithm is used in the SSA for Spectrum Analysis.

D. CAPABILITIES (SPECTRUM)

The capabilities of the SSA are now summarized. Digital sampling is done at baseband frequencies while retaining all passband information. There is no time ambiguity in that all frequency components existed throughout the sampling period. As a comparison, the present FSM uses an analog spectrum analyzer with a scanning display so that each frequency component is sampled at a different time with possible several seconds between start and completion of scan. This method can produce misleading presentation, especially when using Time Division Multiple Access techniques in the satellite channel assignments with access time measured in seconds.

If blocks of data are processed, the size of the block determines the frequency resolution achieved while performing

the FFT process (this is the case with the SSA). Spectrum Analysis can be thought of as presenting the time domain signal to a bank of narrow-band filters each of which is tuned to a different frequency. Resolution is limited by the bandwidth of the individual filters. Each filter is Δf hertz wide (equation 2, Section C) and will vary with N and the sample rate. The result of the FFT is accurate at multiples of Δf or where the spectral component occurs at the peak of the individual filter. Any frequency component that is not a multiple of $\frac{\Delta f}{2}$ then occurs somewhere along the skirt of the filter down to a minimum corresponding to a frequency centered between two adjacent filters. Typically, this error is a maximum of 4.2dB for rectangular weighting. This has the appearance of a "picket-fence" which refers to an apparant amplitude error vs frequency for periodic signals. Translating this to the SSA, the maximum number of samples, N, per block is 2Ø48. The highest sample rate for ADC is 4 MHz. Therefore, a worst case 2kHz frequency resolution can be expected (2kHz = 4MHz/2Ø48). Aliasing can occur if the signal chosen is larger than $1/\tau$. The signal being analyzed must be filtered to the desired spectral bandwidth. The spectrum receiver has this capability. Presently, a frequency resolution of 500Hz can be realized when examining the 1MHz satellite bandwidth. For individual channels a block size of 256 and a sample rate of 1ØkHz are used to give a frequency resolution of 4ØHz.

III. COMPUTER CONTROL OF THE SATCOM SIGNAL ANALYZER

A. INTRODUCTION

The purpose of the SATCOM Signal Analyzer (SSA) is to provide the user a means of doing spectrum monitoring continuously and permit a method of making changes in the spectrum processing with ease.

The control of the SATCOM Signal Analyzer is the result of the integration of the hardware and software into one system. This is only the first iteration. Developing operational concepts is evolutionary. In that the process is evolutionary, what is presented at this time is the SATCOM Spectrum Analysis System as it exists. Modification to the basic programs are contained in Appendix V.

First, some concepts that were in mind during the program development are presented. The software involved with the SATCOM Spectrum Analysis is considered to be part of a system. The systems architecture is the process of partitioning a large software system into smaller pieces. These pieces have a variety of labels such as programs, components, sub-systems and levels of abstraction [17]. This software system represents a set of solutions to a set of distinct but related problems. The distinction between programs and system will rely on intuition, because there is no precise definition. What will be presented will be the objectives of the system, an overview of how the objectives are accom-

plished and lastly, an examination of the set of distinct solutions to the problem of spectrum control.

B. OBJECTIVES

The objectives of the SATCOM Signal Analyzer are:

1. To receive operator parameters from a device.

2. To perform spectrum analysis on a given signal based on the user's parameters.

3. To produce the results in graphic form.

In addition the system must be device-independent (as much as possible) and therefore portable.

C. ACCOMPLISHMENT

These objectives have been accomplished. The accomplishment has been achieved by development of four components. Figure 3-1 demonstrates their perspective. The components are, information in, control of devices, process of data, and information out. The processing of data and information out components have been documented in [16]. Evolutionary programs to replace those in [16] are still in progress. The latest tests of these programs are contained in Appendix V. The process of receiving information and then performing control are grouped into one program.

D. SPECTRUM CONTROL STRUCTURE AND ORGANIZATION

1. Introduction

The program Spectrum Control (SPECTRL) has the objec-

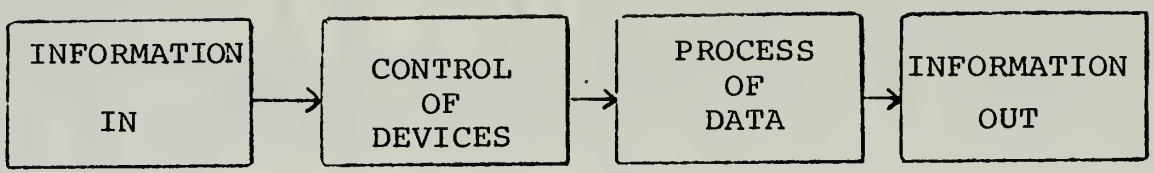


Figure 3.1

SATCOM Spectrum Analysis System: Components

tives of (1) receiving information from a user via a device, (2) performing control of different devices, and (3) translation of information received. Information used in this section means control information vice signal information. The device that is normally used to input information to the system is the C4 panel. The C4 panel is shown in Figure 3-2. Control of the spectrum receiver, local oscillators, and status devices (C4 panel and status display CRT) are performed in this program. Where as control parameters for the AP-12ØB, Tektronix 4Ø14, and DAU are passed to the components doing the processing and displaying of data. The Spectrum Receiver is shown in Figure 3-3. Splitting the actual control up has been done for two reasons. In the case of the Tektronix terminal and the AP-12ØB, no control actually takes place, but rather parameters are passed. The parameters are either indices for looping or reference points for graphics on the Tektronix screen. In the case of the DAU, it has been found that setting all the required registers just before acquisition is prudent. But extensive experimenting in this area has not been done. It might be shown that present acquire routines could be split up. That is, the different registers of the DAU could be set at different times. The translation of the control information is all done in SPECTRL.

Figure 3-4 shows the linkage between the various programs of the SATCOM Spectrum Analysis System. SPECTRL

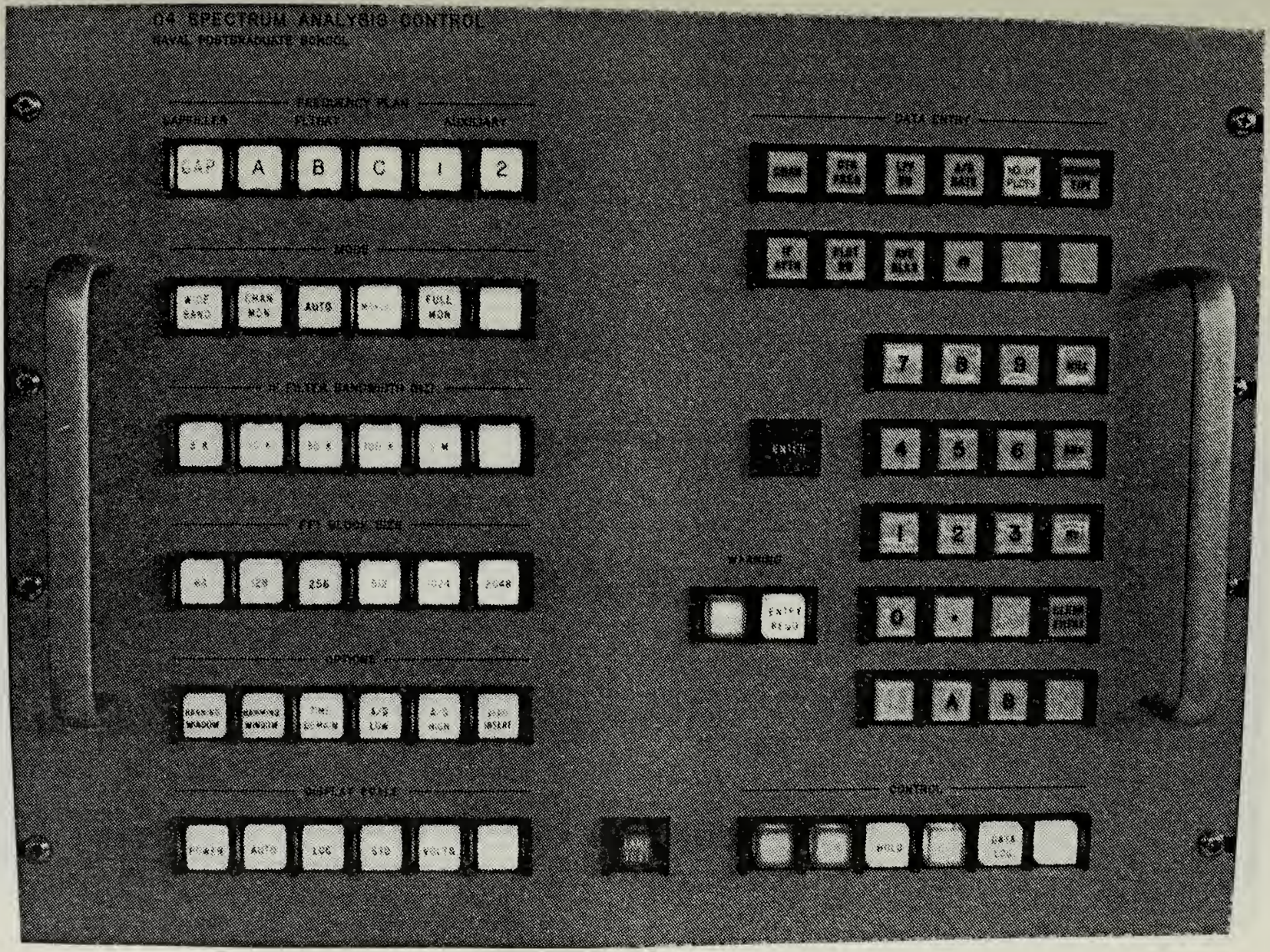


Figure 3.2
C4 Panel

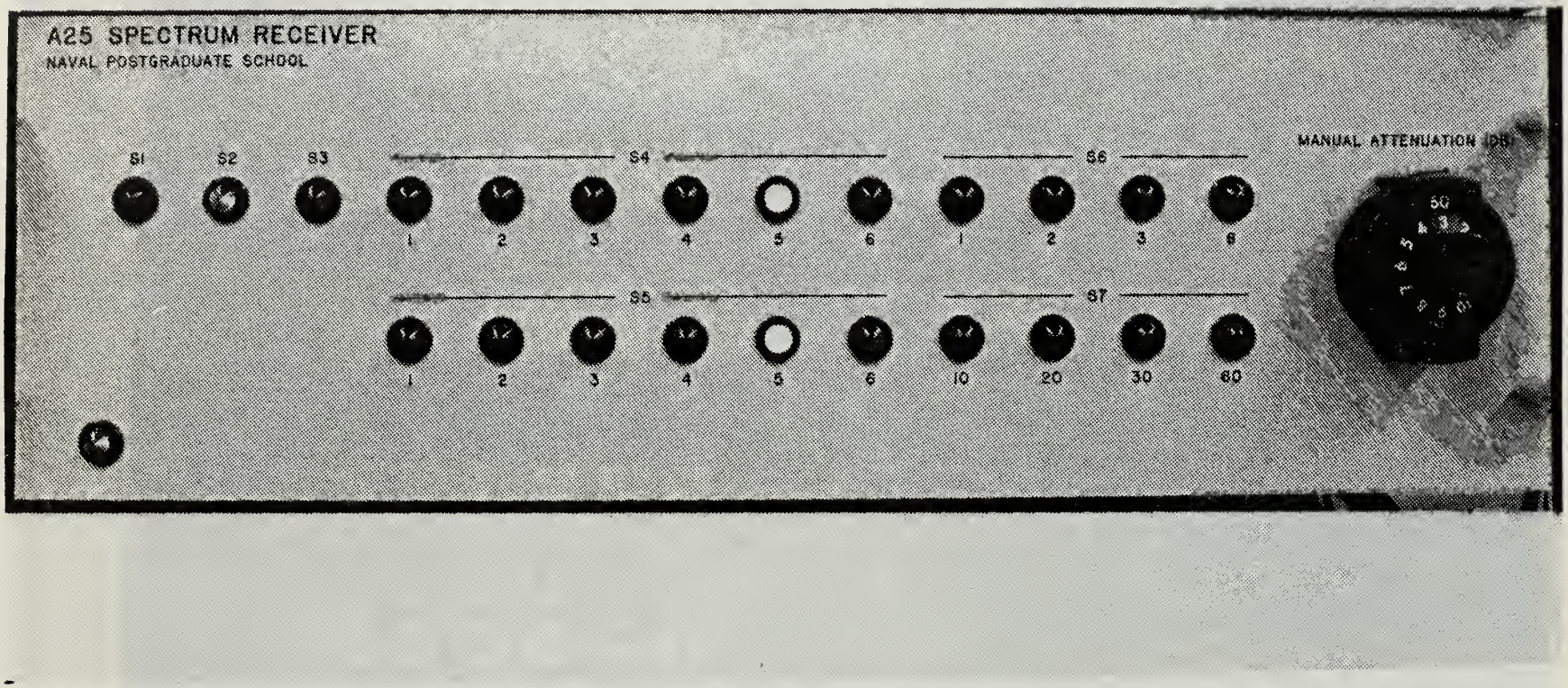


Figure 3.3
Spectrum Receiver

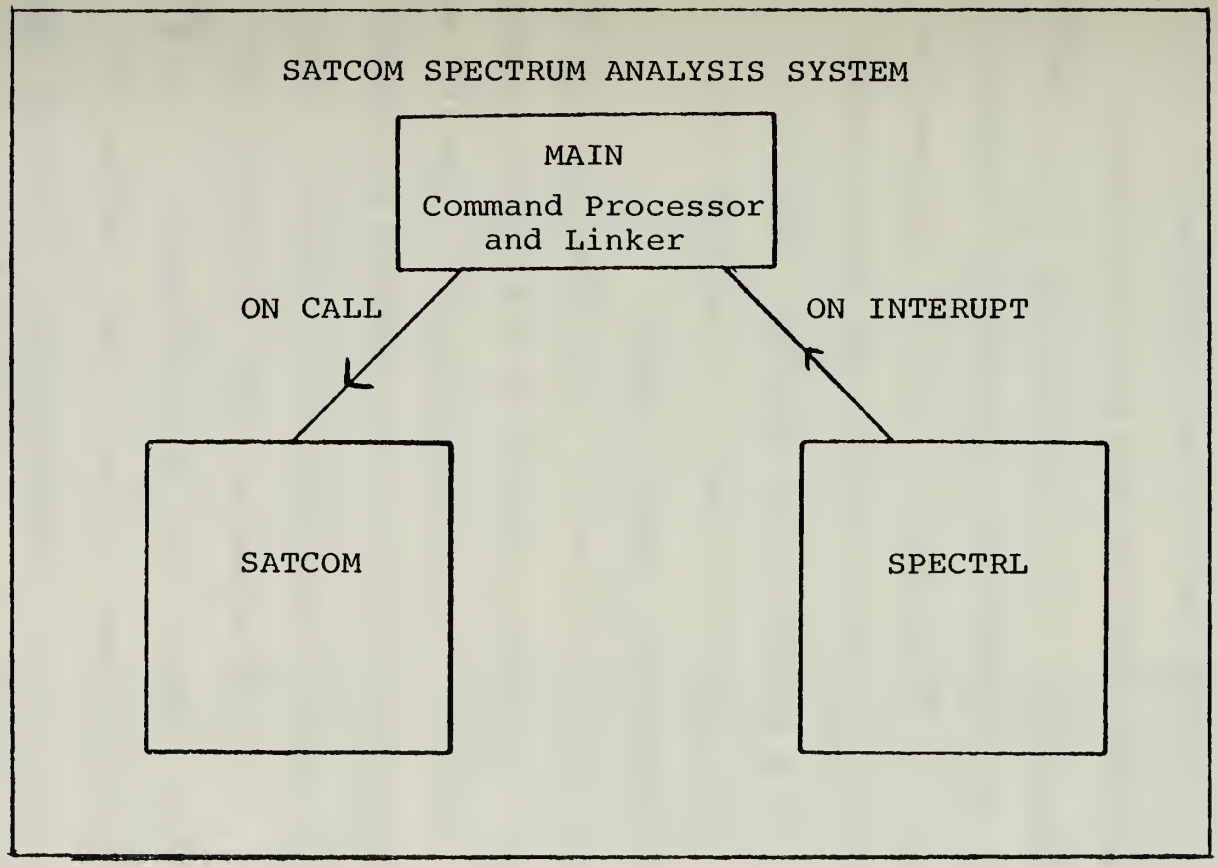


Figure 3.4

SATCOM Signal Analysis System: Linkage

provides the control information and performs control. On conditions set by commands the control information is passed as parameters to the SATCOM program by the program MAIN. MAIN is a command processor and linker. SATCOM presently controls the DAU, processes the data and performs the I/O. At this level only three programs have been mentioned - MAIN, SPECTRL, and SATCOM. (Initialization is done in MAIN). It is significant that SPECTRL can interrupt at anytime to provide information. A discussion of the two programs not yet documented follows. Although under the heading Spectrum Control and not a subprogram of SPECTRL, MAIN is an integral part of Spectrum Control and is presented first.

2. Command Processor and Linker - MAIN

The purpose of MAIN is to (1) initialize the SSA for spectrum analysis, (2) inform the user of the current status, (3) respond to interrupts from the C4 panel, (4) operate on commands, and (5) pass information to the SATCOM program. This is done by two major subprograms and a number of small subprograms. The two subprograms are LINK and BLOCK DATA.

At this time, Table I must be introduced. It is a listing of the variables used. Later is this section the variables will be mentioned. Introducing the variables when first mentioned and stating their purpose will confuse the reader. Instead, all variables are introduced in Table I, and the reader is encouraged to refer to this table in order to follow the documentation.

TABLE I

VARIABLES FROM PROGRAM SPECTRL

Variable Name	FORTRAN Format	Description
LGTAB (78,1Ø)	I*4	An array used for the purpose of a dynamic legal entry table. The columns of the array represent the legal state which the panel is in. The rows of the array represent a button number. If the value of LGTAB (I,J) is 1, then the pushing of the button will not be in error. IF LGTAB (I,J) is a zero then the pushing of the Ith button will cause entry into the illegal button entry handler. The J index is changed depending on which buttons were pushed. For example, the pushing of "CTR FREQ" will cause the state to change so that only the number pad buttons can be entered. In this way dynamic lockout is achieved.
IDEV	I*4	Used to represent the device number in the KTL routines. The spectrum receiver IDEV is 5 for example.
NUM	I*4	Used to represent the number of the button on the C4 panel that was pushed. NUM for RUN is 72.
DECMPT	LOGICAL	Used to set a flag so that decimal fraction numbers can be entered from the number pad. It is set true when the "." (decimal point) is pushed on the number pad; otherwise, it is false.

TABLE I (con't)

Variable Name	FORTRAN Format	Description
NUMBER(16)	I*4	An array used to decode the buttons in the number pad, i.e. a-look-up-table.
NUMREG	I*4	Used to store the integer value of the number being entered via the number pad. It is updated each time a number is pushed and cleared on the "ENTER" and "CLEAR ENTRY" buttons.
PT	I*4	Used to locate the position of the decimal with respect to the number of number buttons that have been pushed.
POWER	I*4	Used to raise the number register to a power in base 1Ø. It is set to Ø, 3, 6 by the "Hz", "kHz", "MHz" buttons.
PONTR	I*4	Used as a stack pointer for the legal state.
JFREQ	I*4	Used as the column index for IFREQ. JFREQ is the selection of the frequency plan 1 for GAPFILLER 2 for FLTSAT A, 3 for FLTSAT B and 4 for FLTSAT C. It is set by the frequency row on the C4 panel.
REMBER	I*4	Used to hold the value of NUM when more information is required. Example - the pushing of IFATN means that a number between Ø and 127 is going to be entered via the number pad. The Legal Table (LGTAB) must change to permit the entry of numbers from the number pad.

TABLE I (con't)

Variable Name	FORTRAN Format	Description
		After these numbers are entered they have to be linked to the IF attenuation control. This is done by remembering which button number was pushed. REMBER accomplishes this.
INDEX	I*4	Used to convert the value of REMBER for use in a computer GO TO.
IIFREQ	I*4	Is the frequency in tenths of hertz for the Fluke synthesizer - passed via the KTLWR subroutine.
LPFBW	REAL	Is the bandwidth for the low pass filter bandwidth of the Rockland Filter passed via the KTLWR subroutine.
IFATN	I*4	Is the value of the intermediate frequency attenuation of the spectrum receiver passed via the KTLWR subroutine.
IFREQ(2Ø,4)	I*4	Is an array of the frequencies used in the FLTSAT and GAPFILLER satellites. The columns are the frequency plans and the rows are the individual channels. Note in GAPFILLER that Narrow Band A and Narrow Band B satellite channels have channel numbers 21 and 22 respectively.
CMD	I*4	Is used to pass a common value of run, reset, or continue to the main program.
HOLD	I*4	Is used to pass the Hold command to the main program.

TABLE I (con't)

Variable Name	FORTRAN Format	Description
STACK(1Ø)	I*4	Is the stack used in changing the legal table. It will permit 1Ø different states that C4 could be in. Presently, there are only four states and nesting of the states has only been down to a second level.
MODE	I*4	Is the column index to the array BUTLIST. It is used for the purpose of creating different sets of button combinations that are to be "pressed"(i.e. simulated) by the computer. The chief application here is to set up modes other than "Wide Band", "Channel Monitor", and "Manual". The Automatic Mode uses this mode index.
BUTLIS(18,6)	I*4	Is an array of button numbers used by the subroutine KTLSIM to simulate the user pushing bottons by the computer. The columns are the modes (i.e. "Automatic Mode 1") and the rows are the buttons to be pushed. The first entry in each column is the number of buttons that are to be "pushed" +1, because the simulation of the pushing of these buttons is done in a do loop.

All of the above mentioned variables are in common with the main program.

ICNT	I*4	Is the variable that contains the value of the block size; used by the DAU, AP-12ØB, and I/O package.

TABLE I (con't)

Variable Name	FORTRAN Format	Description
IADC	I*4	Is the variable used to select high or low ADC in the DAU.
ITIME	I*4	Is the variable used to select a time domain plot (either YES or NO).
IAVG	I*4	Is the variable used to determine how many times data will be acquired and processed, and then this number, IAVG, will be the factor used in determining the average value.
IPLOT	I*4	Is used to indicate the number of three dimensional plots desired.
ILNLG	I*4	Is used to select either linear or log plot in the I/O package.
ICHN	I*4	Is used to select the channel frequency from the array IFREQ and is used to annotate the graph.

The above variables are passed from SPECTRL to LINK. On command LINK will pass these to SATCOM. There were system problems that precluded using these variables in common.

The following variables are used in SPECTRL alone.

FNUMRG	REAL	Is used as the floating number of NUMREG.
STATE	I*4	Is used as the column index in the array LGTAB in order to change legal states.

There are NO other significant variables to MAIN, LINK or SPECTRL.

BLOCK DATA is a program that will permit the initialization of data. There are three variables that should be examined. They are the frequency plan (IFREQ), the legal table (LGTAB) and NUMBER. If a new frequency plan is implemented, the IFREQ array need only be changed. If a different panel or a different input device is used only LGTAB and array NUMBER need be changed. It is hoped that the other entries in BLOCK DATA are self documenting. No flow chart is necessary.

LINK is the core of the command processor and linker. The overall function of LINK is to clear all devices by use of subprograms; set the initial parameters by assignment; inform the operator of the present status by writing to the status display CRT; and wait for a button to be pushed on the C4 panel. The pushing of a button on C4 will cause an interrupt. The response will be to go to the SPECTRL program. The only control information passed to LINK that is acted on by LINK is a command. Other control information has been "passed"; in that it is in common with LINK. LINK will continually check to see if the button pushed was a command button. If it was a command, LINK will act; if not, LINK will keep looping.

The commands are RUN, HOLD, CONT, and RESET. RUN will cause the execution of the program SATCOM. HOLD will permit the storing in memory of the values of the different buttons pushed in an array BUTLIS. This will continue until

the command CONT (continue) is given. On receiving the CONT command, either the program SATCOM will be executed, or if HOLD was previously pushed, the array BUTLIS is used to simulate the pushing of the bottons on C4. In less complicated language HOLD permits the user the ability to set up for another experiment while one is already in progress. CONT will cause the computer to repush all the buttons and then execute the program SATCOM. RESET will cause LINK to reinitialize the program. This has been an attractive feature in this development system. But much time has been spent on its merits in a follow-on-system. The thought has been that one may not want the operator to initialize the process. Figure 3-5 are the flow charts for MAIN and LINK.

3. SPECTRAL

Although looking lengthy and possibly complicated, the SPECTRL program only has four functions. They are to check if the button pushed was legal, go to a location in SPECTRL based on the value that the button represents, perform a particular function, and then exit. The pushing of a button on the C4 panel will cause an interrupt. The interruption will activate the KTL routines which will determine which button was pushed and pass to the LINK program the number associated with the button just pushed. NUM is the variable by which this number is represented. NUM is available to SPECTRL because it is in common with LINK. Information is derived from the value of NUM. This information

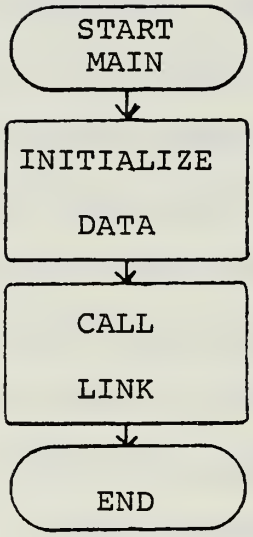


Figure 3.5

Flow Diagram For Main and Link

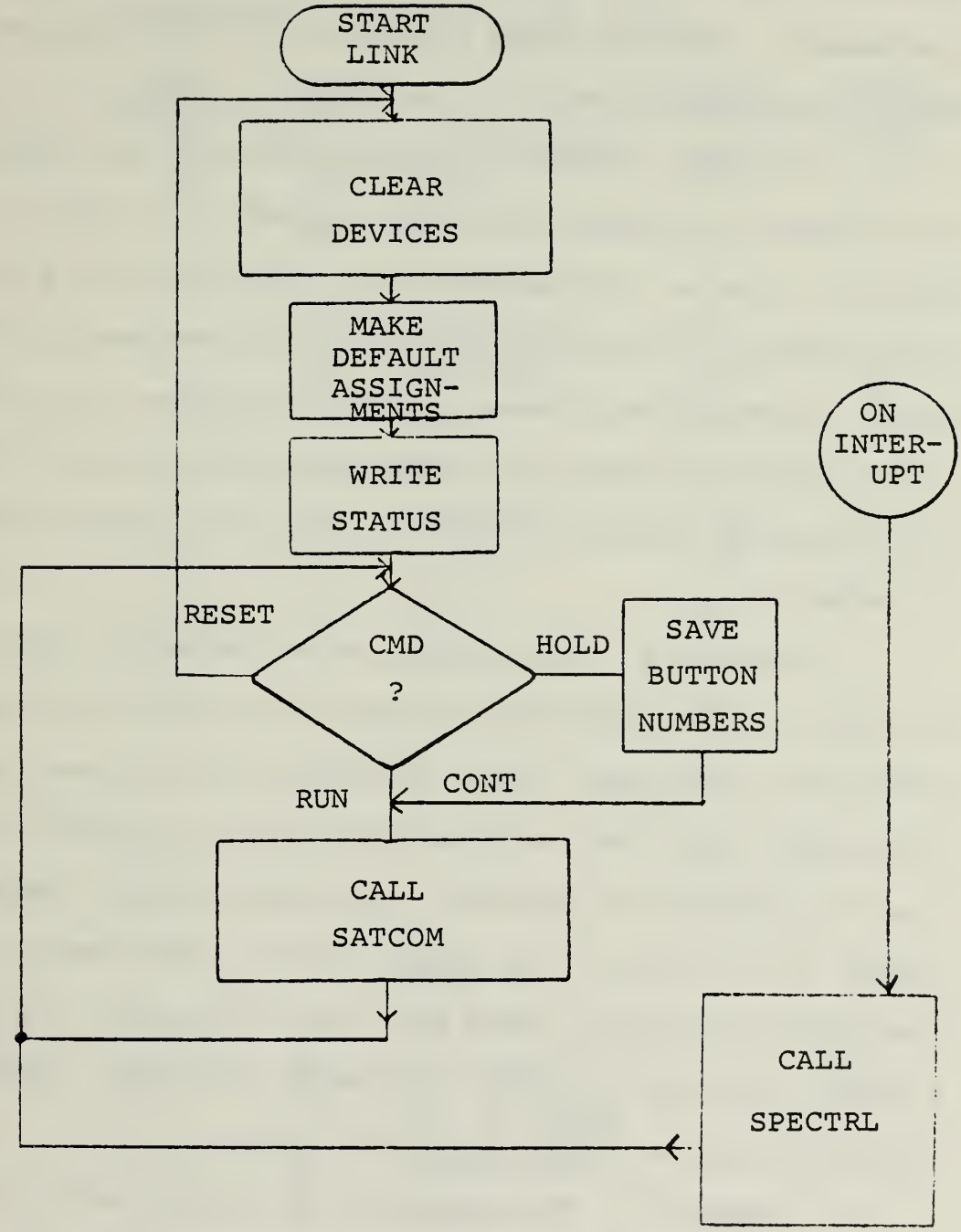


Figure 3.5(con't)

Flow Charts for Main and Link

is used to perform the first three functions which are now discussed. Table II shows the value for NUM and the lamp matrix that corresponds to the button on C4.

In order to provide some protection for the operator and program, a scheme had to be created to be able to dynamically "lockout" certain buttons. This lockout protection is designed to allow the operator who knows very little about the spectrum process to run the system. The dynamically changing legal state will only permit certain buttons to be pressed either because of a certain state or because of sequence.

Example 1: The pushing of a frequency plan button, say GAP for the GAPFILLER frequency plan and the WIDE BAND button for wide band (1MHz) monitoring will place the panel in a state such that only buttons from the CONTROL and MODE rows are able to be decoded. Any other button pushed will result in an error. The error is not a programming error. The SPECTRL program lights the error light and requests another entry by lighting the "ENTRY REQUIRED" light. No other warning or instructions are given.

Example 2: The pressing of the MAN button for manual mode will change the legal state of the panel such that all buttons except those associated with number pad (number Ø-9, decimal point, MHz, kHz, Hz, clear entry) are able to be pushed resulting in a legal entry. In the manual mode it is hoped that the user does know what to do.

TABLE II

LAMP MATRIX AND NUM VALUE TABLE

BUTTON NAME	LAMP MATRIX CODE	VALUE USED BY NUM	BUTTON NAME	LAMP MATRIX CODE	VALUE USED BY NUM
RUN	Ø	72	1ØØK	35	16
CONT	1	73	1M	36	17
HOLD	2	74	(BLANK)	37	18
RESET	3	75	WIDE BAND	4Ø	7
DTA LOG	4	76	CHAN MON	41	8
(BLANK)	5	77	AUTO	42	9
POWER	8	31	MAN	43	1Ø
AUTO	9	32	FULL MON	44	11
LOG	1Ø	33	(BLANK)	45	12
DTD	11	34	GAP	48	1
VOLTS	12	35	A	49	2
(BLANK)	13	36	B	5Ø	3
HANNING WINDOW	16	25	C	51	4
HAMMING WINDOW	17	26	1	52	5
TIME DOMAIN	18	27	2	53	6
A/D LOW	19	28	CTR FREQ		38
A/D HIGH	2Ø	29	LPF BW		39
ZERO INSERT	21	3Ø	A/D RATE		4Ø
64	24	19	No. OF PLOTS		41
128	25	2Ø	WINDOW TYPE		42
256	26	21	IF ATN		43
512	27	22	PLOT BW		44
1Ø24	28	23	AVE BLKS		45
2Ø48	29	24	(BLANK)		46
3K	32	13	(BLANK)		47
1ØK	33	14	(BLANK)		48
3ØK	34	15			

TABLE II (con't)

BUTTON NAME	LAMP MATRIX CODE	VALUE USED BY NUM	BUTTON NAME	LAMP MATRIX CODE	VALUE USED BY NUM
7		49	.		65
8		5Ø	(BLANK)		66
9		51	Clear Entry		67
MHz		52	(BLANK)		68
ENTR		53	A		69
4		54	B		7Ø
5		55	C		71
6		56			
KTz		57			
1		58			
2		59			
3		6Ø			
HZ		61			
ERROR	22	62			
ENTRY READ	23	63			
Ø		64			

Example 3: The pressing of any of the DATA ENTRY buttons will change the state so that only buttons in the number pad will be accepted. Advisement will come up on the status display stating that a number must be entered via the number pad. The "ENTRY REQUIRED" light will also light, because data are required.

The legal table has a different state for each of the modes available. There is an additional state of allowing only the number pad to be used. The dynamism of the legal state is handled by the concept of a stack and pointer. The legal state can change to ten different states with this pointer. The states are pushed onto and popped from the dynamic stack. The first state in is the last state out.

Going to a particular location in SPECTRL is accomplished by a computed "GO TO". In many cases the value that NUM holds is usually the first two digits in the label that the computer "GO TO" will be vectored. Example: The value NUM takes on for the RUN button is 72. The computer GO will vector the program to label 72ØØØ. This was done for ease in reading and making the button association with hardware easier.

The performance of a particular function is the core of the SPECTRL program. This particular function can be characterized by (1) making an assignment, (2) performing control, and (3) making a calculation. In some cases more characteristics are present.

Example 1: The pushing of GAP for a frequency plan change only changes a column index in the frequency array IFREQ.

Example 2: The pushing of WIDE BAND makes assignment that will be passed to the DAU Control registers, and sets the receiver up on the proper frequency with the proper bandwidth and attenuation.

Example 3: The pushing of any button in the block size will cause the block size to be computer from value of NUM which is related to the button pushed.

In essence, that is the structure of the SPECTRL program. Flow diagram for Spectral is shown in Figure 3.6.

E. SPECTRUM CONTROL - OPERATION

1. Introduction

The execution of these programs will permit one to operate the SSA for spectrum monitoring. The loading and execution of these programs and any other programs are discussed in Chapter IV and Appendices F and G. Operating the SSA for spectrum monitoring is not difficult. The parameters can be changed while processing is going on to set up for another experiment. Assuming that the program has been started, the C4 panel lights will all be turned off and an instructional message will be printed on the status display panel. Initializing all the devices will have been done. The program will be in a loop waiting for a button

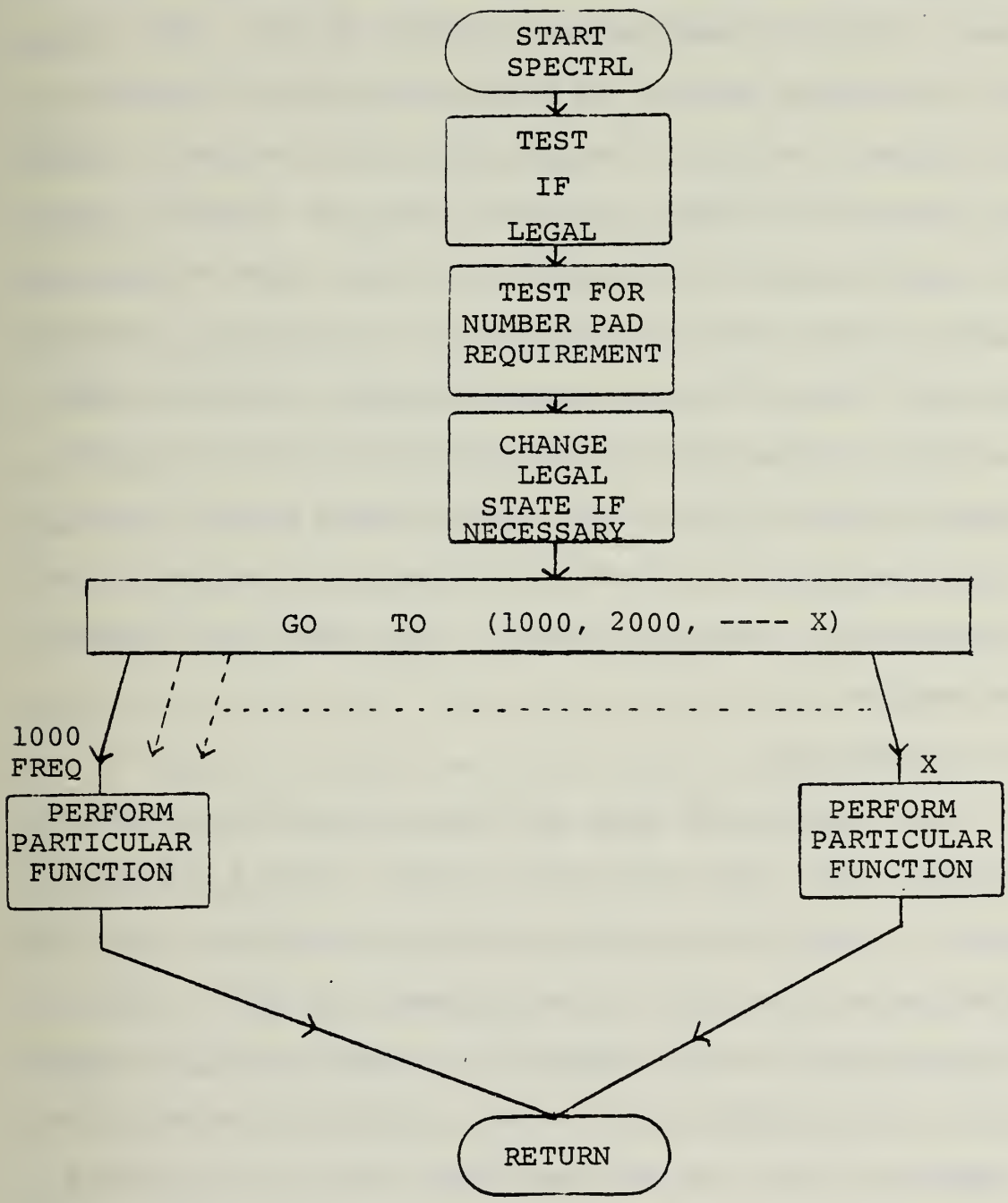


Figure 3.6

SPECTRL Program Flow Diagram

to be pushed. Operating the spectrum process requires at a minimum, a mode of operation and command to run. The sequence for pushing buttons is not restrictive if a manual mode is chosen, but it is restrictive for other modes. The layout design of the panel suggests that the operator start at the top left and proceed down the panel, then to the top right and proceed down to the control row, giving a command as the last item. There are defaults set if certain modes are selected. The defaults were designed so that the user would be protected if the user did not make an entry and also as a convenience. These are modes that require a number of buttons to also be pressed. But these are "pressed" by the computer.

2. Selections

It is suggested that the first selection be that of a frequency plan. The Auxiliary buttons 1 and 2 are not programmed. The next selection would be a mode of operation. Wide Band mode will set all the equipment up for a 1 MHz spectrum analysis on the GAPFILLER Frequency Plan (Channels 1 and 2). All defaults will be set and only a button from the CONTROL or MODE row will be legal entries. To change any of the defaults (which are indicated on the panel by lighting of buttons that the computer "pressed") press MAN for manual operation. In chosing CHAN MON for channel monitor also sets up standard defaults. But unlike Wide Band monitoring the program needs to know which channel is

desired to be monitored. The legal state will change to permit only entries from the number pad to be legal entries. The operator will only be able to enter a number (use channel numbers 21 and 22 for Narrow Band A and B of the GAPFILLER Frequency Plan). Once the channel number is entered the program will pick the correct channel frequency from the chosen frequency plan and change the legal state so that only a mode or control command can be given. AUTO is for automatic mode. It has been programmed to the extend that it operates as CHAN MON in requiring a number to be entered (e.g. AUTO 1 for automatic Mode 1). But the particular functions for the automatic modes have not been defined. Comments in the source listing indicate where these functions can be defined. MAN selects the manual mode. The pushing of this button will change the legal state such that all buttons (except the number pad) can be pressed. No defaults are set. FULL MON button is still in development. Complete revision of the program STACOM are required for this evolution. These programs and documentation are in Appendix V. FULL MON is for a full monitor of the GAPFILLER Satellite. It is a monitoring of the Wide Band and two narrow band channels. This is accomplished by setting the spectrum receiver and DAU up for a signal at a particular frequency; acquiring data; process the data, and; plot and hold the data on the graphics terminal. This is then repeated again, but the receiver and DAU parameters are changed. The control of

the spectrum receiver, and DAU has been completed. The I/O graphics package is complete. The processing of the data is currently under the second iteration and not available.

All other selections can only be made if the manual mode has been selected by the pressing of MAN. Selection of the buttons in the IF FILTER BANDWIDTH row will change the spectrum receiver bandwidth. The indications that this has been accomplished are: (1) the button lights up, (2) the operator can hear the filter change, (3) the front panel lights to the spectrum receiver will change, and (4) the HP-122Ø scope presentation (which shows the down converted signal) will change. A selection from the FFT BLOCK SIZE row will change the block size. The indication that this has happened will be only the notation on the plot, and the light in the button pushed will be lighted. Three buttons in the OPTION row are under development. They are the windowing buttons. No windowing has been implemented. TIME DOMAIN will cause the system to display a time domain graph of the base band signal. A/D HIGH and LOW will select the high or low speed analog-to-digital conversion process in the DAU. Only two buttons in the DISPLAY SCALE row are programmed. POWER will cause a power or linear spectrum to be displayed when the data is graphed. LOG, however, will cause the data processed to be plotted on a log scale. In the DATA ENTRY rows only PLOT BW button is not programmed. These two rows provide a unique feature. They will permit the entry

(via the number pad) any number for any of the categories represented by the button. All buttons operate in the same manner. For example: the pressing of IF ATN for IF Attenuation will cause the legal state to change to permit only entries from the number pad. A number (Ø-127 in this case) can be entered by pushing the numbers in the number pad. After the numbers are pressed, press the ENTER button. The ENTER button will change the legal state back to what ever state the panel was in at the time IF ATN was pushed. The pushing of ENTER will also cause the number just entered to be used to control the spectrum receiver attenuation. This same sequence will apply for all the buttons that require a data entry.

The number pad is only engaged when any button pushed has a particular function which requires a numerical entry. CHAN, for example, when pushed, will engage the number pad, because by itself Channel means nothing. Channel 5 means something. CTR FREQ for center frequency means nothing but center frequency 15ØMHz means something. The Clear entry will clear all registers associated with the number pad. ENTER button will clear the number pad, "pop" the legal state off the stack and make the number just entered available to the particular function that requested the number pad.

All the buttons on the control row are commands to the MAIN program. The pressing of RUN will cause the execution of the SATCOM program. HOLD will permit entries on

the C4 apnel but nothing will take effect until CONT is pressed. RESET will cause the program to start again. DATA LOG has not been programmed.

IV. COMPUTER OPERATION OF THE SSA

A. INTRODUCTION

The purpose of this section is to document procedures for operation of the SSA using the host computer. To do this, a number of items have to be presented. Many of these items are discussed in the manufacturers' manuals. But in an effort to decrease the "start-up-time", i.e. the time it takes one to get accustomed to the system so that more time is spent doing experiments rather than debugging operator problems, some of the pertinent concepts and commands will be presented with examples. In addition, due to the nature of the SATCOM lab, the transient students, and the ease with which diagnostics can be run, diagnostic procedures are presented. It is felt that anyone can run any diagnostic at any time. This should reduce troubleshooting-time by localizing a problem, reduce repair costs, and increase familiarity and confidence with the system.

Operation of any of the devices by the computer requires a program and the device to be available. The approach will be to go through each device and state how each one is made available. Particular eccentricities of the device will also be made known. Having presented the devices, then a discussion on programming the system is given.

B. DEVICES

1. Power

All AC power is distributed through the main switchboard located at the entrance to the lab. The power is distributed from a polyphase system of three phases. Each phase differs by one-third of a cycle of 120^{o}. The phases are labelled on the switchboard as phase A, phase B, and phase C of which C has four branches.

a. Phase A

Phase A distributes power to racks 7-11, 15-19, and the HP 983Ø calculator. All of these racks require power when running the SSA.

b. Phase B

Phase B distributes power to computer racks 12-14. Although the DAU is resident in rack 14 it is able to be powered up after rack 16 is powered on.

c. Phase C

Phase C has four branches to it. For computer operations only branch 4 and one switch in branch 1 to the AW/WSC-3 has to be on. Branches 1-3 provide power to the lab benches, rack Ø-6, and uninterruptable power to the time code generators and frequency reference amplifiers.

2. Unit Power

Refer to Figure 5-1 for unit location in the racks.

a. Racks 7-11

Most items in racks 7-11 are powered up when

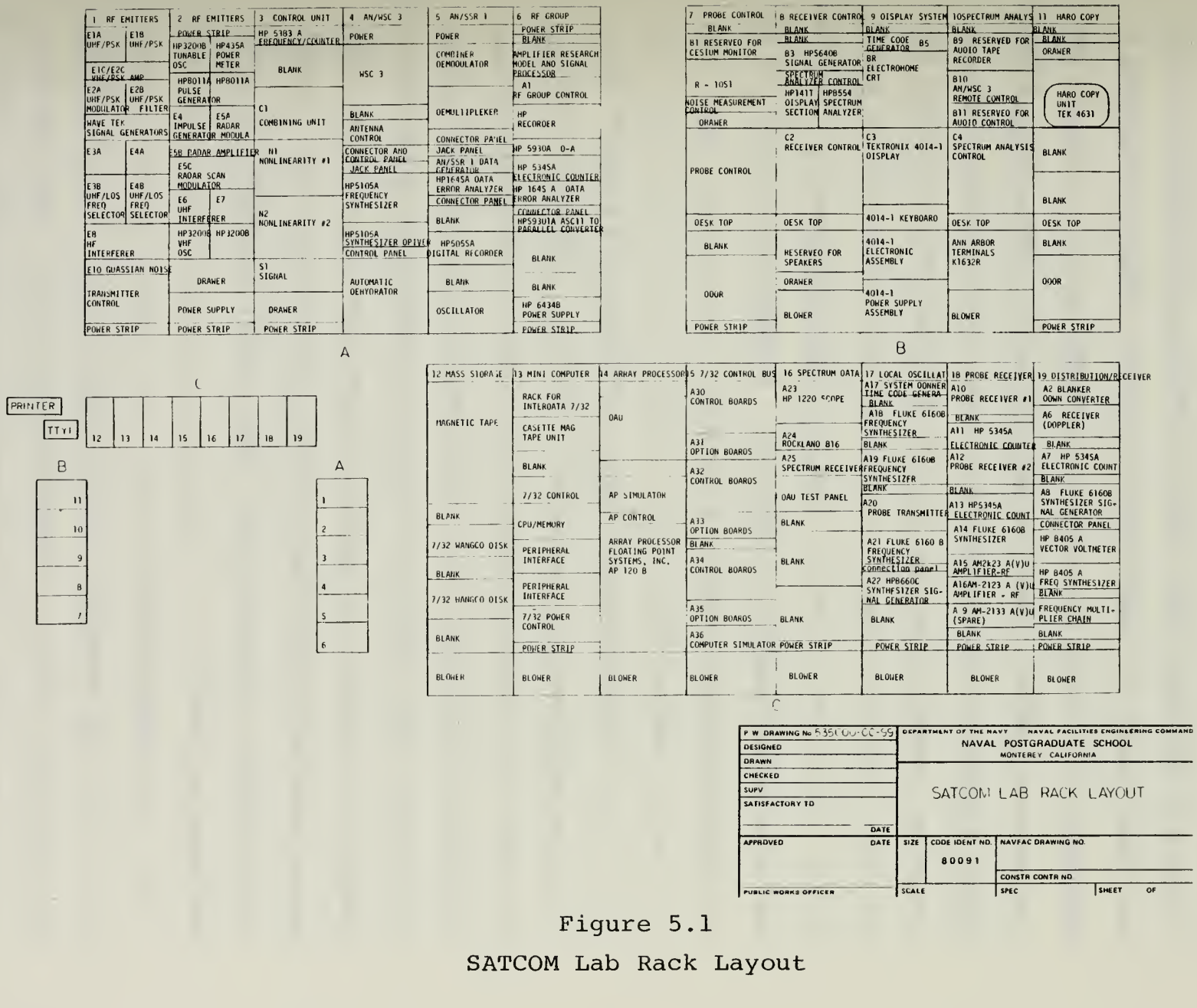


Figure 5.1

SATCOM Lab Rack Layout

phase A switch 1 and 2 are thrown. The additional considerations are the Tektronix 4Ø14-1 terminal, the Tektronix 4631 hard copy unit, the Electrohome display, and the Ann Arbor terminal. All these devices are powered up only when in use. The Tektronix 4Ø14 terminal must have the "Reset page" button pushed before powering down this unit. This same button should be pushed after the screen "blooms" when powering on. The hard copy unit takes 5-7 minutes to warm up before reliable copies can be made. The Ann Arbor terminal is located just above the blowers in rack 1Ø. It must be powered on for the Electrohome CRT display to work.

b. Racks 16-19

All devices in these racks will be powered up once the switches on phase A are engaged. Note that the HP 983ØA will also be powered up or down if the power to rack 19 is on or off. The DAU power supply is located in rack 16 and this condition should be recognized when securing the DAU or rack 14. If a unit device does not come on after powering up racks 16-19 then look on the device for a local on/off switch. There are no local on/off switches for the NPS made equipment.

c. Racks 12-15

The powering up and down of rack 15, the control bus interface boards, is detailed in /¯18_/. In rack 14, as has been stated, the DAU resides, but its power supply is connected to rack 16. The AP-12ØB is powered by local on

and off switch and the rack 14 switch of phase B. Racks 12 and 13 house the computer, magnetic tape and disc drivers. The power to these units is always left on. The procedures for powering up the disc drives are set forth in Appendix F. The magnetic tape drive systems are all powered on by the "power" switch on each unit. The computer is powered by a key switch. In the lock position the switch panel is disabled.

d. Devices not in racks

The printer, teletype, and console/center table get power from phase C, branch 4. The high speed printer not only has to be powered up, but also must be in the run mode. This is achieved by simply pushing first the ON and then the RUN buttons. But to advance pages manually one must press STOP and then press the paging buttons. The OFF button secures power to the unit. The teletype is powered on by the local power switch on the lower right. If the teletype is unavailable after is has been powered on, then examine the fuse to interfacing power supply located in the teletype near the right side. This fuse has a history of failing when power to the computer is lost and then again restored while at the same time the teletype did not get powered down. The console and all units on the center table also get power from phase C, branch 4.

3. Procedures for Power Up

Detailed walk through procedures for powering up the

SSA are contained in Appendix F.

C. PROGRAMMING THE SSA

1. Introduction

The information contained in this section is to provide the operator sufficient information to quickly operate the system. It is not meant to be a "cook-book". The danger of such an item is that the operator will only be able to perform specific procedures. It will be impossible to perfrom other tasks without some concept of how the system works.

The type of operation that is performed on the computer has either the nature of a one word command such as to execute a program that already exists; or to write, perform language translation, and execute one's own program. The most common causes of errors on this system are due to the operator not knowing which program is running and how the computer system refers to a file. One concept should be fundamental, and that is: anything that runs on the SSA computer is only a program. Most students that use this system are familiar with other computer systems that are larger, and seemingly everything (loading a program, assignment of logical units, and execution of that program) was done automatically - "transparent to the user". If start up time for the student on this system is to be reduced, the student must be able to identify which program is running, and at which level it is running. Too much time can be

spent looking in the wrong error list only because one did not know which program gave the error. An example of confusion would be one in which the FORTRAN compiler gave an error message and because this message appeared on the system console the operator might mistake it for an operating system response.

What will be presented will be pertinent features and or commands of the following areas

a) the operating system

b) CSS commands

c) utility programs (editor, language translation, disc utilities)

d) diagnostics

e) specific project programs

f) special device oriented programs (AP-12ØB, HP 983ØA).

2. Operating System

The name of our operating system is OS/32 MT. It is a standard, general purpose operating system for Interdata 32-bit computer systems. It supports a multiprogram environment with up to 255 user tasks written in FORTRAN, COBOL, BASIC or ASSEMBLER languages. An important feature to new operators is the Command Substitution System - CSS. The primary reference /19/ is quite adequate, but should not be taken rigidly. (Some of the SSA addresses are different as well as other features.) The following are sections that should be read and experimented with during the operator's

first session on the computer:

Chapter 3 sections: 3.1 - 3.3
3.4.3
3.4.5
3.4.13
3.6.1 - 3.6.9
3.6.12
3.7 - 3.8

Chapters 1, 3 - 5, and 9 should be read after having been exposed to the system.

The MT in the name OS/32MT, stands for Multi-Tasking. The noun, "task", is a confusing label in the reference manual. A task is any job that is running, paused or just loaded in memory. It could be the users program, the editor program, or any other program. An example follows. A user would make his program, for example, a program written in FORTRAN, a task by first creating this program using the editor. Then this FORTRAN language program would be converted to assembly language and then to object form. The object form ("deck"/module) is then established into a task. It is only in this form that this user's program can be executed. During this whole process an editor program, and compiler, and assembler programs had been running as tasks in order to do the process. The last program to be running was the Task Establisher program running as a task. Understanding what a task is is important, because in order to

know which program was running is equivalent in knowing what was the task. The fact that the OS/32 MT is multi-tasking does not mean it is a time-sharing system. There is one system console, and this is not shared. The significance of this is that there is only one point of control and is restrictive in that nature. But it is also very powerful. The amount of control the user has must be realized, for the user could do damage to his or anyone else's files or running programs. If the system is not time-sharing but is multi-tasking what is the difference? The operating system will permit 255 tasks to be operating at one time. This means one user can be using the systems editor while creating a program while a second user is collecting data. This is all done by partitioning memory between the tasks.

The OS/32 MT supports the concept of a background partition and a foreground partition. Only one job or task can be loaded in background whereas several can be loaded in foreground. Any job running as a background task is prevented from corrupting the tasks in foreground. On the SSA system program development is done in background while SSA experiment programs or data collection programs are running in foreground.

How the operating system does what it is to do is now presented. The purpose for this is because one must be aware of (1) how a user written program is linked to the device, in order to program the device or (2) how to run more

than one program that communicates with one particular device. It is also necessary to be aware that there has been considerable software development at the operating system level to link user programs that communicate with devices on the control bus.

To place things in perspective the components of the operating system are:

System Manager
Executive Functions
Task Manager
Timer Manager
Memory Manager
File Manager
I/O Subsystem
Resident Loader
Floating Point Support

The components that are of significance for operation of the system and understanding of the software development are the System Manager and the I/O Subsystem. The System Manager is of interest to the general user primarily. The System Manager handles all interaction between the system and the system console device. It provides the operator interface to the OS/32 MT. It executes as a task in OS/32 MT, and is designed so that many functions are performed through Supervisor Calls (SVC). The System Manager contains routines to support direct access devices. All I/O requests to the

system console and logging device are controlled by the System Manager. Commands from the system console are decoded by the System Manager, and then it calls the appropriate executor. It contains logic to provide the console operator with messages in case of error. The I/O Subsystem is composed of a SVC 1 handler, the peripheral device drivers, SVC 11 handler, and other routines such as the System Queue Handler. This section is of interest to users who will write drivers for future devices or for those who desire to run SSA devices on other host computer systems. All devices that are connected to the control bus are naturally driver dependent, but also they are dependent of the SVC 11 handler and KTL library.

To see how all the programs fit in a system will aid one in software development. Figure 5.2 is a simplified software system diagram. Two partitions are made in this diagram to represent views from which one can understand the software system. These partitions are User/Executive and levels I, II & III. The distinction between the levels is basically one of language and specifics. The lower the level, the lower the level of the language and more specific is the instruction. The distinction between the User and Executive is not as clear. In the sense used here, executive denotes administration and decision making for the computer system. The User however, is an employer who employs the services of

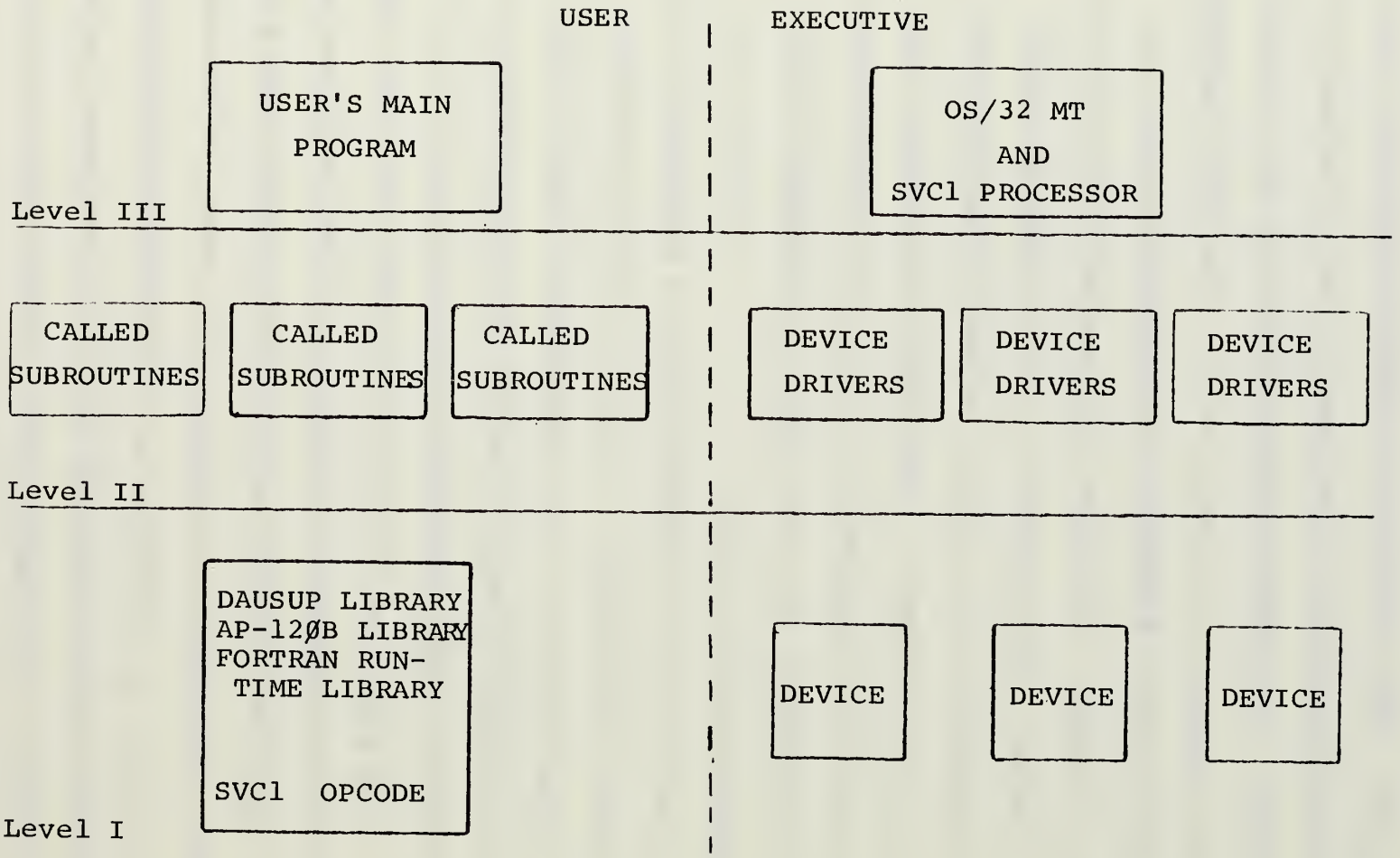


Figure 5.2

Basic Diagram of the Software System

the Executive. The link between the partitions is the SVC - service request call. There are a number of SVC available and one doing program development should be aware of them. They are as listed in Table III. (Note that the documentation for all Interdata software makes no reference to SVC 4, but in the source listing this SVC is used. Also, SVC 14 is available to the USER as is 8, 1Ø, 12, 13, but SVC 14 is shared with the OS/AIDS program.) Lastly, in this figure only the SVC 1 is shown as being linked to the Executive.

An explanation of what actually happens is helpful in understanding the software components. The approach will be top down. The user's program in this example will be a student written program in the FORTRAN language and has the task to add two numbers and print the results on the printer and then plot the point on the TEKTRONIX terminal using the graphics capability. The numbers are to be added in the AP-12ØB. The program first will READ the data in from the console. Assume that the following is the correct FORTRAN program.

```
      .
      .
      .
      READ (5,6) A, B
      .
      .
      .
      CALL VADD (A, 1, B, 1, C, 1)
      .
      .
      .
      WRITE (6,16) A, B, C
      .
      .
      .
```

TABLE III

OS/32 MT SUPERVISORS CALLS

SERVICE			FUNCTION
1			General Purpose I/O
2	CODE	1	Pause
		2	Get Storage
		3	Release Storage
		4	Set Status
		5	Fetch Pointer
		6	Unpack
		7	Log Message
		8	Interrogate Clock
		9	Fetch Data
		1Ø	Time Wait
		11	Interval Wait
		15	Pack Numberical Data
		16	Pack File Descriptor
		17	Mnemonic Table Scan
		18	Move ASCII Characters
		19	Pack
		2Ø	Expand Allocation
		21	Contract Allocation
		23	Time Management
3			End of Task (EOT)
5			Fetch Overlay
6			Intertask Services
7			File Management
9			Load TSW
11			NPS Control Bus Device Dependent I/O
14			User SVC
15			ITAM Device Dependent I/O

```
    CALL DRAWABS (A, C)
    .
    .
    .
```

The above program is the USER MAIN PROGRAM. Subroutines from the AP Library (VADD) Tektronic Library (DRAWABS) and FORTRAN Library (READ & WRITE) are used. The AP and Tektronics library are one level down from the USER MAIN PROGRAM and one level above the FORTRAN run time libaray (RTL). For this example we will assume all I/O was done by subroutines in the FORTRAN Run Time Library, and further they were done by a SVC 1. The READ statement causes an entry into the FORTRAN RTL to subroutine READ. This causes the execution of the assembly language op-code of SVC. In executing that op-code certain parameters (i.e. the data, the logical unit, etc.), are passed to the operating system. The operating system processes the parameters and makes the connection of logical unit number 5 and console (the logical unit having been assigned before executing the program). The operating system passes the information to the console driver. The driver sends information to the console. The console's hardware generates something on the screen that meant "give me something". Upon entering a number (two in this case), the data would return via the driver to the SVC 1 processor, and the data would be placed in memory addresses A and B. Note that the administrative function of getting the data and storing it was done via the OS/32 MT SVC 1 processor. The

WRITE statement is similar. It causes the SVC 1 processor to get data from memory. Location B; decodes the logical unit and sends data via the software driver to the device - the printer. VADD and DRAWABS are only intermediate steps on the USER side. It is important to note that in all cases every level on the USER side is device independent. Even though subroutines to the AP-12ØB and TEKTRONIX were used it was assumed that all I/O was to be done via the SVC 1. This was not a great restriction, for there are many programs written for other devices, i.e. DAU Support Library, that use the SVC 1 as its link with the operating system. How are they device independent if the program is designed for a particular device? The programs are device independent in that they will take input from any device and although its' output is designed for a specific device, the output could be directed to an alternate device and the system will still run - i.e. no crash. All user programs should be written as device independent programs. This must be done so programs can be transferred from one system to another, and so that new replacement devices can be added, (without having to change the user program), as they will be in this development system. What changes then? The executive side. All the drivers are specific to their devices. The idea is to keep things general and system oriented at the USER level and push the specifics to the lowest level and if possible all the way to the drivers. Two examples

follow. Consider changing only the Fluke frequency synthesizers. Only one driver need to be changed presently. But if one had written a user program that was tailored to the frequency synthesizer, then not only would the driver have to be changed but also the user program. The second example exhibits not the change of a device but rather the change of a bus either the control bus or the IEEE 488. It might not be necessary to change the drivers to all the devices on the bus, but rather create a SVC designed for a particular need. This is how the present SVC 11 was developed. If the user programs are dependent on developemnt devices and busses then there will be a lot of reprogramming as the devices change.

Before leaving the operating system, the one major stumbling block in the mechanics of operating the system is a misunderstanding of what a file is. On this system a file can be almost anything. A file does not mean just some software module on a disc. The printer is a file, and the teletype is a file. There is a file called the "NULL:" device. By making a logical unit assigned to the NULL device output requests are ignored. No output takes place. That is, when the operating system processes the SVC1, no output takes place. The program will continue. An example for use of the NULL: device follows. In software development a user has a program that is to print out a number many times before ending. To have this same program run to check

out just the logic of the program the NULL: device could be assigned to that logical unit to which the printer might normally have been assigned. Files all have names to them. At all times all components of the name are used by the operating system, but it might not be necessary to type the full name either due to a default or the way a CSS is written. All devices have file names that end with a colon (:). By typing D D on the system console one can see the labels that exist for the devices in the lab. Note the "NULL:" device. Specifically, the following commands could be given assuming assignment of logical units had previously been done.

USER TYPES	MEANING	COMPUTER RESPONDS
D LU	Display Logical Units	Ø5 CON Ø6 PR
CL, 6	Close Six	
AS 6, NULL:	Assign Six to NULL:	
D LU		Ø5 CON Ø6 NULL

The three components to the file descriptors are volume, file name, and extensions. File descriptors apply to the software files on the disc. Example: MON 7: SPRCVR.FTN. This is a FORTRAN file with the name SPRCVR on the disc MON 7. The extensions that are used on our system are located in the reference manual [17]. The added extensions are listed in Table IV. The error associated with the file descriptors are

1) Not including the colon (:) when referring to a

TABLE IV

AP-12ØB FILE DESCRIPTOR EXTENSIONS

.APS - AP SOURCE

.APO - AP OBJECT - NOT 7/32 OBJECT

.APE - AP EXECUTIVE - FOR AP SIMULATOR

device.

2) Forgetting that the system volume was being used as the default.

3) Not knowing that a CSS command was written taking into account that an extension would not be given.

For clarification of the problems read the section on CSS commands.

3. CSS Commands

The Command Substitution System (CSS) is an extension to the OS/32 MT command language. It is nothing more than another program. The statements in this program are either operating system command or other CSS commands. Instead of typing a number of operating system commands to do the job all these commands can be placed in one file called a CSS file or program. When this program is executed, one is executing a CSS command. Note that unlike operating system commands, CSS commands are user written programs and, therefore, changeable. These "commands" can be created by the use of the editor or "BUILD" command. The obvious application for the CSS commands are those when the user does not want to issue a set of operator commands to do a particular job waiting for each result of the individual operator command. As an alternate: group in one CSS program all the commands that could normally be issued Execute the CSS program. The amount of time that can be saved in doing this is probably the single most useful feature of the

system to students that use it. Certainly there are many features to the SSA, but these are expected. To not have to "baby-sit" the computer will free the student's time.

How to write these CSS commands is well covered in /17/, but here is the concept of what happens. The operator will type something on the console. The computer will then try to decode it as an operator command. If it cannot be decoded as an operator command then it will treat the something as a file. It will then go to the file, read the file and try to execute what it is reading, as if the file was a set of operator commands. If the file does not exist then an error will be displayed. Examples follow.

Example 1: Assume the user types "PR:". PR: is not an operating system command; so, it will check to see if PR: exists as file, and it does. Remember devices are "files" and PR: is the label for the printer. The computer will then go to the printer and try to read; it cannot, so an error will result.

Example 2: Assume a file exists with the name JOB.ASN; that the statement in that file is: "AS 5,CON:"; that the user has a job he has loaded as a task and now would like to assign logical units. The user could type "AS 5,CON:" using the operating system command to assign logical unit number 5 to the console. Or one could type JOB.ASN. This is not a operating system command so the computer looks for the file JOB.ASN, reads it, and tries to execute it, which

it can do in this case.

Example 3: Assume the user wants to make five object modules from FORTRAN sources; wants to make one task called JOB.TSK; has a TET file to include the five object modules; wants to copy this file to a cassette tape, and then display the devices on the system. To do this by operating system commands would take the same time to do it as with a CSS command, but with the CSS command the user does not have to wait to execute the next command. The user would enter the editor and create a CSS program, and save it in file with a label of his choice - assume the label for this file is "DOIT.CSS". The actual entries would be:

BIGBOY ONE, TWO, THREE, FOUR, FIVE	-	CSS command to FORTOBJ files ONE, TWO, THREE, FOUR, FIVE. FORTOBJ is a CSS file to take (file name). FTN and language translate it to (file name). OBJ.
FORT JOB	-	a CSS command to language translate and include all files in a TET file to make a task called (file name). TSK
COPYT JOB.TSK, CAS1:	-	a CSS command to use OS COPY to copy a task from file 1, to file 2 where file one is JOB.TSK,

file 2 is CAS1: in this case.

D D - a operating system command to display devices.

$EXIT - a CSS statement that indicates this is the end of the CSS command - DOIT.CSS

Note this CSS command is just another program, where the statements are other CSS commands and operating system commands. It was created by the user for a particular use. Now, by executing DOIT just by typing "DOIT" the computer is executing the commands in DOIT rather than executing the commands from the console. (The console has not lost control.) For detail use see /19/. A number of CSS commands were provided by Interdata, C3, and BDM. Some are also provided in this report in Appendix U.

The problems that arise in using CSS commands are in two areas, but are associated with the descriptor and how the system uses it. The file descriptor includes: volume, file name, and extension. To execute CSS commands the CSS file must be either on the "system volume" (set by the "V" command) or the full file descriptor must be used. Assume the system volume is MON7, and the CSS file, DOIT, is on MON5. To execute it one must either type MON5:DOIT or change the system volume to MON5. The second problem that

usually occurs in the execution of a CSS command is that the computer will use the system volume as a default if it is not specified. In some CSS files the extensions are already provided. Example: "FORT JOB.FTN" typed on the console will give an error, because the passing parameter is JOB.FTN. Reading what the FORT CSS command is will show that it has replaced an "@1" with the phrase JOB.FTN. But note the very next character to the @1 is .FTN (@1.FTN). So the resulting descriptor is: VOLM:JOB.FTN.FTN.

System Volume	JOB.FTN	FTN	- used by the computer
(default)	@1	FTN	- written in the CSS

The computer will not permit double extensions (FTN.FTN) and the result will be an error. Now let us assume that JOB.FTN is on MON5, that the system volume is MON7 and FORT is the CSS file on MON5. To execute FORT one would type MON5:FORT JOB. The system volume will be used as the volume for the file descriptor and the FORT command will cause the computer to look for a file MON7:JOB.FTN to compile because the default volume is the system volume, and the computer will never find it. To avoid these problems know what the system volume is and change it if need be. Know what the passing parameters are to the CSS programs. Know what extensions the CSS program will add on, if any. Always remember that the system uses a full file descriptor by adding a default if the user does not give a full file descriptor.

4. Utility Programs

The purpose of this section is just to expose the various utility programs that are available. In keeping with the terms expressed in the previous section, the utility programs are User programs. This might have been confusing to one who has just skimmed the operating system reference manual [19], because good descriptions of these programs are contained there. The implication might have been that they are part of the operating system - they are not. The utility programs have such a label because they exist to help the user create an executable program. First, there is the editor which is normally loaded and started by a CSS command. This will be the most used method for creating any files, hereafter called source. (The other method is with the "BUILD" command - see [19].) After creating a file with the editor a number of language converters, translators, compilers, etc., are used to get the source in a form that can be loaded into memory and then executed. The language utility programs that are supported are BASIC, FORTRAN, COBOL, APAL (which is assembly language for the AP-12ØB), and an assemblier language. To build the loadable module or program the TAST ESTABLISHER TASK must be run. This is just another program which is loaded like any other program which takes as inputs an object program and commands and provides as output a loadable task (i.e. program). Discussion on the use of these programs is provided in Appendix G.

There are utility programs that are useful for keeping one's disc clean and backed up. The programs are DISCHECK, DISCDUMP, DISCINIT, C3COMP, BACKUP, and OSCOPY. DISCHECK is used primarily when (1) the operating system crashed while a disc was marked on and (2) a session with the editor or the execution of language translation did not come to normal termination. It is used after these occurrences, because there will be files that have been "left open". DISCHECK will close them. DISCDUMP is used to dump files from a disc to another disc, magnetic tape or the printer. It can be used to back up a disc. DISCINIT is used to rename a disc or completely clear the disc of all files. The clearing of the physically lowest fixed disc in the SSA system is regularly done. It is done to make the lowest disc available for temporary files. Discs are also cleared before a compress is done. C3COMP is the compress program. After a period of time free sectors on a disc will become scattered. It is more efficient to pack all the free sectors together. To use the free spaces up, the programs must be compressed. C3COMP will do this. There are two files with the file name BACKUP. One is a task; the other is a CSS command. The CSS command will backup one's disc on another disc and a magnetic tape as well as do a compress on itself. It is extremely useful to keep disc clean, compressed, and backed up. It is faster to backup a disc on a magnetic tape than it is to copy selective files to a tape.

The BACKUP program and its use is adequately described in [19]. OSCOPY is primarily executed with a CSS command. This program is used to transfer ASCII, binary and established task (ones with the extension ".TSK") files from their source to any other file destination. To copy a ASCII device such as the printer will cause the printer to act on the binary code which was being sent. Since it wants ASCII code, the printer might have passed wildly a whole ream of paper before the user realized the mistake.

There exists also a software debugger program called OS/AIDS. To use OS/AIDS the object module must be included when making a task. That is to use the debugger on a program called JOB, then the TET commands or TET file must have the command "INCLUDE OSAIDS.OBJ". OS/AIDS is then stated when the program JOB is executed. Certainly the normal application is to debug a program. There is another application that is particularly useful. It is when a new developed device is being interfaced to the system. OS/AIDS will permit one to stop the program at almost any point and permit the examination of any of the memory locations and registers.

5. Diagnostics

Diagnostics are run on a device when the device is not operating correctly. There are diagnostics for most devices. Table V provides a listing of the devices, location of the diagnostics for the device, if the operating system

TABLE V

DIAGNOSTIC LISTING

DEVICE	LOCATION OF DIAGNOSTIC	OPERATING SYSTEM DEPENDENT	DOCUMEN-TATION
CPU	Interdata Tape & Disc	NO	YES
MAC	Interdata Tape & Disc	NO	YES
Memory	Interdata Tape & Disc	NO	YES
Logical Storage Unit	Interdata Tape & Disc	NO	NO
Universal Clock	Interdata Tape & Disc	NO	NO
Ann Arbor Terminal	No Diagnostic	--	--
Cassettes	Interdata Tape & Disc	NO	YES
Printer	Interdata Tape & Disc	NO	YES
Extended Selector Channel	Interdata Tape & Disc	NO	NO
Magnetic Tape	Interdata Tape & Disc	NO	YES
Universal Logic Interface	Interdata Tape & Disc	NO	YES
Disc	Interdata Tape & Disc	NO	YES
Current Loops	Interdata Tape & Disc	NO	YES
Tektronix Graphics	Tektronix Tape, Cassette, Disc	YES	YES
HP983Ø	HP System Cassette Tape	NO	YES
Control Bus	Control Bus Cassette Tape	NO	Appendix A
AP-12ØB	AP-Tape, Disc-MON1	YES	NO
DAU	DAU Diagnostic Cassette	YES	YES

must be up to run the diagnostic, and if there is documentation for the diagnostic. Diagnostics are simple to run. The essence of running diagnostics is to load the program, start the program, and compare the results. In Appendices H through T are the procedures to run the Interdata provided diagnostics. There are step by step procedures to load, start and compare the diagnostic process.

The Tektronix graphics are run with the operating system by use of the CSS command "RUN TCSDIAG". The output for the different tests differ with those shown in the Plot 1Ø Verification Routine Users Manual, Release 1, Tektronix, Inc., Beaverton, Oregon. Enter 3 when asked terminal type and enter 2 for buffer type. For character sec enter 96Ø, 154ØØ, or 3Ø7ØØ for 9.6 K, 154 K, 3Ø7 K band respectfully.

For the HP 983Ø simply load the system test tape and follow instructions in the Model 3Ø Calculator System Test Instructions manual.

To test the control bus see Appendix A.

Testing of the DAU is done by a series of tests titled DAUTEST, DAUTEST1, and DAUTEST2. These are on MON7 and are run by use of the GO CSS command.

Revision two software has been received from Floating Point Systems. The new tests for the AP-12ØB have not been run as of this writing. There is no documentation for the revision one tests other than source listings.

6. Specific Project Programs

In this development system programs are continually being created, revised and deleted. Because only eight characters can be used in naming files, and because the names are quite arbitrary not much about a program can be gleamed from the name of the program. The solution to this problem does not lie in reading previous thesis, but rather providing future users one source for all programs that have been considered useful for present and future uses. This one source has been established. It consists of a file which contains the listings of all the general purpose programs. BDM services at the end of each quarter have been copying the unique files of graduating students to a master tape.

These listings will serve two purposes. First, they will reduce the duplication of effort. For example, the individual who is trying to do something and needs a good I/O package for graphics, such a package has been made and available. Communication with the DAU is in fact done by a group of programs that are in the DAU support library. All data acquisition through the DAU uses these programs. These programs might be subprograms or subroutines. The second purpose these programs have has more to do with changing the system. Insight can be obtained if software is to be replaced by hardware or vice versa. The nature of all the programs in the lab are either processing, interfacing (con-

trol and I/O) or diagnostic in nature. To this date most programs are device independent except diagnostics. Some interface programs are written for the control bus (SVC II and KTL routines). If, for example, the interfacing is to be changed by replacing a bus structure with a microprocessor, the essence of the bus can be seen in the interfacing program. If a different processor is used, the algorithm and many modules of a original program will assist in writing a new program. The only changes that are expected are the called programs.

The specific project programs are all operating system dependent (or run on the HP 983Ø alone), and written in the SATCOM lab. A diagnostic program exists for the data acquisition unit. Processing programs exist for FFT, scaling and averaging in the AP-12ØB. They are the first iteration and welcome changes. I/O programs exist for plotting data on the Tektronix. Interfacing programs exist for the control bus and HP 983Ø but the HP 983Ø has problems with the operating system. Programs also exist for interfacing with panels C1, C2, C4 and devices A25, A26, A18, A19, A21, A22, A1Ø-A14, A6-A8. See Figure 5.1 for SATCOM lab rack layout.

The SVC11 and KTL libraries have been provided by BDM. Documentation for these routines is contained in the listing. Their use and purpose will be briefly stated here. The KTL routines perform I/O on the control bus. KTL is

mnemonic for control. The routines will permit writer, reads, status request, enable interrupts, simulate interrupt, return from interrupt, connect and disconnect to a device, and wait for task trap. All KTL routines are FORTRAN callable. Table VI lists the KTL routines. The SVC11 handler processes the SVC11 calls made in the KTL routines. It is used instead of the normal I/O calls for the control bus devices in order to eliminate the overhead of normal I/O calls which might be characterized as being a nuisance for devices which do not transfer large amounts of data. SVC11 incorporates INT11 and other logic to handle control bus device interrupts almost independently of the I/O systems.

The DAU Support (DAUSUP) Library was provided by Lieutenant Gary W. Bohannan. Documentation for these routines are contained in Table VII. These routines provide communication with the data acquisition unit in the samy way that all other I/O is done. It has been observed that the subroutine DAURUN should be preceded by a command to the prestart circuit in the DAU. This is done by use of the DAUREG routine. A program example is provided in Appendix U. It has not been determined that this corrective action is absolutely necessary. First starting the prestart circuit and then calling DAURUN was a solution to a problem that occurred after the DAU had been operational for about four months. The DAU has been run since without the fix and operated without problems. Nevertheless, it is suggested [18]

TABLE VI

KTL ROUTINES

Arguments: IUNIT - Integer number of the device

IVALUE - Integer number and value sent or received

VECTOR - Address vector (lable in FORTRAN Program)

Vector = A'1ØØ' for FORTRAN IV

or = NAME - External for FORTRAN V

*KTLWR (IUNIT, IVALUE)

Performs a write operation to device IUNIT writing IVALUE to that device.

*KTLSIM (IUNIT, IVALUE)

Simulates an interupt to device IUNIT, with junk, IVALUE.

*KTLRD (IUNIT, IVALUE)

Performs read operation from device IUNIT, stores the read value in IVALUE.

**KTLON (IUNIT, VECTOR)

**KTLOGG (IUNIT, VECTOR)

Connects and disconnects device IUNIT to a program and physically enables interrupts.

KTLST (IUNIT, IVALUE)

Gets status from device IUNIT.

KTLWT

Puts the task in a wait state (like pause)

KTLRET

Returns from Interrupt

* function value = SVC11 status

Ø = Okay

TABLE VI (con't)

1 = illegal unit number

-1 = device off line

4 = illegal value for a write

** Function = Svc 6 Status

Ø = Okay

Anything else see /-17-/.

Note that C1, C2, C4 panel buffers bounce badly. Any new devices that bounce as badly as the panels and for these panels the programmer would be prudent to disconnect the device for about 1ØØ μsec immediately after an interrupt has been detected. Then re-connect the device before doing a KTLRET.

TABLE VII

SUBROUTINE LIBRARY DAUSUP

The following FORTRAN callable routines are provided:

```
BCDFRQ (FF)
SETSYN (ID, IR)
DAUREG (IVAL, IREG)
DAURUN (ICNTL, ICODE)
DAUGET (IARRAY, IADDR, NCOUNT)
```

All routines, except BCDFRQ, require INTERGER*2 data elements as arguments. See page 27 of FORTRAN Ref Manual paragraph 6.2. This specifies 16 bit (2BYTE) HALFWORDS compatible with the DAU interface.

NOTE: You may not use literal constants in argument lists, these are, by default, INTEGER*4 values.

EXAMPLE 1:

```
    .
    .
    .
  CALL DAUREG (4, 5)          is not correct form
```

EXAMPLE 2:

```
    .
    .                         correct form
    .
  IX = 4
  IJ = 5
  CALL DAUREG(IX,IJ)
```

Subroutine BCDFRQ requires a REAL (floating point) argument.

In order to attach DAUSUP programs to your main program, an additional line is required in the TET sequence. This is easily accomplished by creating a disk file (using the EDIT sequence) with the filename:

```
name . TET      where "name" is the same
                as the "name" in name . FTN
```

This .TET file should be as follows:

TABLE VII (con't)

```
(Col. 1)
  LOG
  REMOTE
  ESTABLISH TASK
  MXSPACE 28ØØ
  OPTIONS F
  GET  6ØØ
  INCLUDE            <Includes Main Program and your own
                      subroutines.

  EDIT DAUSUP.OBJ       <Here we sat to GET DAUSUP
                         ROUTINES as required

  EDIT TCSLIB.OBJ       <Optional - TEK4Ø14-1
                         Plot 1Ø Programs if required

  INCLUDE APSUP.OBJ     <Optional AP12ØB programs
  EDIT APLIB.OBJ

  EDIT FVRTL.OBJ        <Required for all FORTRAN Programs

  BUILD TASK, name.TSK  <Here again "name" is same an in
                         name.FTN

  MAP

  END
```

SUBROUTINE BCDFRQ(FF)

- FF = SAMPLE RATE, IN HZ, (FLOATING POINT)

- Subroutine converts frequency datum to four BCD digits plus Range Digit for compatibility with the SYNTEST SM1Ø1 FREQUENCY SYNTHESIZER used in the DATA ACQUISITION UNIT FOR SAMPLE RATE CONTROL.

  Calls Subroutine SYNSET to complete output to DAU control Interface for Synthesizer set up.

  Out of range data is reset to nearest limit.

  Language - FORTRAN

SUBROUTINE SYNSET(ID,IR)

- ID(4) INTEGER*2 ARRAY-4FREQ DIGITS

TABLE VII (con't)

- IR INTEGER*2 RANGE DIGIT

- Subroutine completes BCB formating for SYNTEST SM1Ø1 Frequency Synthesizer and outputs to Data Acquisition RANGE and FREQ registers to set up SM1Ø1 to sampling rate for A/D converters. No format checbing is provided.

- MAKES SVC calls to OS for output via DAU driver.

- Requires Logical Unit 1 set to DAU.

- Language - Common Assembly Lang. (CAL).

SUBROUTINE DAUREG (IVAL, IREG)

- IVAL - INTEGER*2 DATA TO BE OUTPUT

- IREG - INTEGER*2 DAU INTERFACE REG #

- Subroutine outputs 16 bit Halfword value to DAU interface register specified by IREG.

- Makes SVC call to OS for output via DAU driver.

- Requires Logical Unit 1 to be set to DAU.

- Language - Common Assembly Language (CAL).

SUBROUTINE DAURUN (ICNTL, ICODE)

- ICNTL - INTEGER*2 control code to select A/D converter pair.

- ICODE - INTEGER*2 status code

  = Ø Acquisition complete

  ≠ Ø Acquisition incomplete, DAU error.

- Subroutine selects DAU interface control register, outputs ICNTL code and starts DAU. On acquisition completion, DAU status is checked and ICODE is returned to calling program.

- Makes SVC call to OS for I/O via DAU driver.

- Requires Logical Unit 1 to be set to DAU.

TABLE VII (con't)

- Language - Common Assembly Language (CAL).

SUBROUTINE DAUGET (IARRAY, IADDR, NCOUNT)

- IARRAY - INTEGER*2 ARRAY for receiving data. DIMENSIONED AT LEAST TO NCOUNT.

- IADDR - INTEGER*2 DAU buffer memory start address for take out.

- NCOUNT - INTEGER*2 Total no. of 16 bit halfwords to be transferred. Usually set to 2* Acquisition count.

- Subroutine outputs IADDR to DAU Buffer Memory Address Register and NCOUNT+1 to DAU Word Count Register. Note that the actual transfer count is controlled by the Selector Channel for DMA transfer and the DAU word count may be set to any value larger than acquisition count for this step.
  The Control Register in the DAU is then set to indicate transfer to the 7/32 and a DMA transfer is set up and run.

- Makes SVC calls to OS for I/O via DAU driver.

- Requires Logical Unit 1 set to DAU.

- Language - Common Assembly Language (CAL).

that the prestart procedures be retained in any data acquisition program. These problems should be considered in the future data acquisition units and corresponding software.

AP-12ØB programs are well documented in the manuals [20 - 24] provided by Floating Point Systems. To easily use these programs a CSS command has been provided. The CSS command allocates the necessary files; starts the program APAL; on completion of APAL, APLINK is started; on completion of APLINK, APSIM is started. Once APSIM is terminated the CSS command is cleared. With this CSS command one can assemble the source program into a machine code for the AP-12ØB and test it on the simulator - all by one command.

The HP 983Ø primary function has been control of devices on the IEEE 488 bus. Programs have already been presented [12, 14, 16] that communicate on this bus. Whatever changes that come to the bus there is one potential stumbling block which can be avoided. All devices on the bus have addresses. Some of these addresses have been set in by hard wiring, others by switches. The switches make it easy to change the addresses. But this should be done with caution, because the probability of having one address for two devices is possible. Table VIII provides the current directory for the IEEE 488 bus. Should these addresses change and/or devices change the reaifications are not only to addressing conflicts but also to past proven programs. "If it can happen it will". Scenario: the bus appears bad,

TABLE VIII

HP-IB ADDRESS DIRECTORY

ADDRESS SWITCHES	TALK ADDRESS CHARACTER	LISTEN ADDRESS CHARACTER	DEVICE	LOCATION	SERIAL NUMBER
00100	D	$	3330B	BENCH	1313A00540
01010	J	*	TIME CODE GEN	RACK 17	#1205
01101	L	,	TIME CODE GEN	RACK 09	#1204
01110/1	N or O	. or /	COUNTER	RACK 19	#17234
10000/1	P or Q	0 or 1	COUNTER	RACK 18	#1612A01791
10010	R	2	3320	RACK 19 (LOW)	
10011	S	3	8660C	RACK 17 (BOTTOM)	
10101	U	5	9830	DESK	
10110	V	6	DVM3490A	BENCH	
11000/1	X or Y	8 or 9	COUNTER	RACK 18	#1624A02137
11011	{	;	8660C	RACK 11	#1643A00963

because an addressing conflict occurred due to a change in equipment where the change was not known to the programmers. A proven program might then be run as a "test". The test will fail. At some time later the addressing conflict will eventually be discovered.

Some problems with the HB IB exist and no solution is apparent at this time. First it is not understood why the remote command (768) must be output on the bus twice to place all the devices in remote. Sometimes issuing the remote command will work. But issuing it twice has always worked on all devices. To use the "VAL" statement to read the HB IB ensures that the "COM" statement and "DIM" statements are well understood. If these are not used correctly the "VAL" statement will not pass the correct value. The use of all statements are well covered in the string variable ROM manual and operating manuals by Hewlett Packard [25, 26, 27]. Due to the format of the output from the time code generators made by Systron Donner the output must be read in segmented strings and not in one string variable. The bus follows strict instructions. It is not sufficient to enable a device in an output mode. The device must also be told to "talk" as in the case of the digital volt meter (HP 349ØA).

V. CONCLUSION

The software portions of the SATCOM Satellite Signal Analyzer documented in this thesis have been developed, and tested successfully. Further software modifications are recommended a) to increase the throughput of the system, b) to allow windowing, and c) to allow greater flexibility in the output.

APPENDIX A

CONTROL BUS AND CONTROL BOARD OPTION 'A' DIAGNOSTIC MANUAL

1. Control Board Option 'A' Test Program

   1.1 Related Documents:

   Test Program

   Universal Logic Interface Instruction Manual 29-311

   1.2 Test Programs to be run prior to loading this test:

Series 32 Basic Test	06-158
Series 32 Processor Test	
Part 1	06-154
Part 2	06-155
Part 3	06-178
Series 32 Memory Test	06-156
TTY Basic Confidence Test	06-004

2. Purpose of Test

   2.1 General Test Information

The Control Board Option 'A' Test Program is used to check the proper operation of the Control Board and Option 'A' board (CBOA). Byte data transfers are exercised. Output commands, Status Request and the interrupt mechanism are also exercised.

   2.2 Test Description

The program checks the byte mode by writing a shifting data pattern that is alternately a One (1) in a field of Zeros (0) and a Zero in a field on Ones. The data written

is read back from the module and compared to the original pattern. The same pattern is used to test the status return and command latches. After data transfers are complete, the interrupt mechanism is tested. Output commands are used to disarm, arm and enable the CBOA and generate an interrupt. The device number on acknowledge is also checked.

3. Minimum Hardware Required

The following is a list of hardware necessary to perform this test:

1. The 7/32 Processor
2. 8KB of Memory
3. Console Input/Output Device (See Appendix B) Teletype or CRT
4. Device under test - CBOA
5. Option Board 'A' Test Connector

4. Requirements of Machine Under Test

4.1 Test Configuration

The option board 'A' test connector must be attached on the output of option board A. The wiring pin diagram is described in Appendix E.

4.2 Device Address

If the control board address is other than X'8F' enter the device address by using the DEVADR option.

4.3 Console Device

If the console device is other than a teletype with device address of X'02' see Appendix B for program modification.

5. Loading Procedures

5.1 Normal Loading Procedures

1. Manually enter the X'50' sequence shown below:

	Location	Contents
	X'30'	X'0000'
	X'32'	X'0000'
	X'34'	X'0000'
	X'36'	X'0050'
	X'50'	X'D500'
	X'52'	X'00CF'
	X'54'	X'4300'
	X'56'	X'0080'
For CAS1	X'78'	X'45A1'
For CAS2	X'78'	X'55A1'

2. Place cassette in CAS1 or CAS2; Power ON; Rewind to load point; Switch to ON line

3. Execute at address X'30' (DTA, 30, ADR, RUN)

   The boot loader will load first

   Hit RUN - File mark passes by

   Hit RUN - Second file mark passes by

   Hit RUN - Program loads - All zeros come up on display. If not, rewind tape; start this step again.

4. Refer to Appendix B and set up the addresses for Console I/O Device if other than teletype.

5. Address location X'A00' DTA, A00, ADR

6. START program execution. Observe that the following title is printed on the console device:

CONTROL BOARD & OPT A TEST

6. Operating Procedures

6.1 Normal Testing

1. Ascertain that OPTION BOARD A TEST CONNECTOR is properly plugged in.

2. When an asterisk (*) is printed enter the desired options via the console device. Refer to Appendix C for the option explanation.

3. Enter the RUN command via the console device.

4. If no errors are detected, characters "NO ERRORS" are printed almost immediately after execution. Should an error occur, refer to section 6.2 for the appropriate action.

5. To re-execute the test, enter the RUN command via the console device.

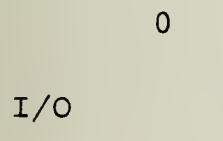
6.2 Error Procedures

If an error is encountered, the Processor loops on the failure, the error number is displayed on D1 of the Processor Display Panel, and the error number is printed on the console device. Refer to Appendix D for the meaning of the error number.

APPENDIX B

CONSOLE DEVICE DEFINITION

1. The halfword labelled I/O (see the listing) has the default value for Teletype (address X'02') as the console device. If the configuration is different, the test program must be changed as follows:



Console Device Identifier	Explanation
X'01'	GDT/CRT on PASLA/PALM Interface, strapped for FDX and the highest baud rate
X'02'	TTY on TTY Interface GDT/CRT on Current Loop Interface
0,X'03-X'FF'	Reserved. The program defaults it to 2.

2. The teletype or Current Loop Interface, if used, should be strapped for the device address of X'02'. If it is different, the halfword labelled TTYADR (see the listing) must be changed accordingly.

3. The GDT (Graphic Display Terminal) or CRT; is used on PASLA Interface, should be strapped for the device address

of X'10' and X'11' for receiving and transmitting side respectively. If it is different, the halfword labelled CRTADR (see the listing) must be changed accordingly.

APPENDIX C

OPTION TABLE

Option	Default Value	Description
DEVADR	X'8F'	Specifies the device address of the Control Board
INTRPT	X'0'	Specifies whether the CB0A interrupts will be tested. 0 = interrupts are tested. 1 = interrupts are not tested.

APPENDIX D

FAILURE NUMBER DEFINITION

NUMBER		FAILURE
There is no error 01		
02	Byte Mode of Byte to Half-word Mode Data XFER	R4 = Data Written, R6 = Data Read
03	Sense Status Bits (0:3)	R4 (12:15) = Test Data R6 (12:15) = Error Bits
04	Output Command Bits (4:7) or Sense Status Bits (4:7)	R4 = Output Command R6 = Returned Status
05	Unsolicited Interrupt	
06	No Interrupt	
07	Wrong Device Number Returned on Acknowledge	R4 = Received Device Number
08	Acknowledge Does Not Reset Attention	
Interrupt Level = X		Where X = Register Set that the Interrupt Vectored into.

APPENDIX E

OPTION BOARD A TEST CONNECTOR

Signal	to	Signal	Pin	to	Pin
SIN 020		DOT 060	26		16
SIN 030		DOT 070	8		34
SIN 040		COT 040	27		15
SIN 050		COT 050	9		33
SIN 060		COT 060	28		14
SIN 070		COT 070	10		32
DIN 000		DOT 000	7		19
DIN 010		DOT 010	25		37
DIN 020		DOT 020	6		18
DIN 030		DOT 030	24		36
DIN 040		DOT 040	5		17
DIN 050		DOT 050	23		35
DIN 060		DOT 060	4		16
DIN 070		DON 070	22		34
SATNO		COT 040	3		15

APPENDIX F

POWERING UP THE SSA

A. OBJECTIVE

The objective of this appendix is to provide procedures for powering up the host computer, Interdata 7/32, the disc drives, and other peripherals. Also included are the steps to be followed to boot load an operating system.

B. CONDITIONS

In following these procedures there are only two decisions that must be made. They are:

1) Is the computer up with a good operating system?

2) Which peripherals are needed?

The condition of the computer being up with a good operating system can be recognized by either all zeros in the hexadecimal display or with date and time displayed in the hexadecimal. The date will show the day and month using two characters each - the order might be reversed depending on the operating system. But the time in hours and minutes is shown always in the last (right most) four characters. If the display has anything else it does not have an operating system in it.

Examples:

bDDMMHHmm - Good operating system
bMMDDHHmm - Good operating system
bCODEnnn - Crash code of nnn - bad system

000000000 - Good operating system

C00000000 - Good operating system

where

b = blank

D = day

M = month

H = hour

m = minute

n = number

CODE = CODE

C. PROCEDURES - POWER UP

1. Computer up, with good operating system (lights across front display either the date and time, or all zeros), if not go to 2.

a. Turn main power switch to console table ON (Phase C, Branch 4, also other main power switches for other peripherals)

b. Turn Super Bee console ON (switch in back of console, lower right, facing the back)

1. Push "RESET"

2. Push "CLEAR"

3. Push "ONLINE" on the keyboard

c. Discs

1. Turn power ON (one or both drives)

2. Wait for "LOAD" light

3. Insert or remove disc pack

4. Close cover

5. If cartridge is loaded, push "RUN"

6. Disc is ready to be marked "ON" in the operating system when the "READY" light is lighted. Note to operate a fixed disc a removable disc must be loaded.

d. Teletype

1. Turn switch to "ON LINE"

e. Printer

1. "ON"

2. "RUN"

f. Tektronix

1. "ON" - lower right on stand

2. After tube blooms, "RESET"

. Only turn on those peripherals needed for a given session

2. Computer down, unreliable operating system (lights across front off, or displaying an error code)

a. Turn computer power ON, phase main switch (first switch)

b. Turn console table power switch ON, Phase C, Branch 4

c. Turn Super Bee ON (switch in back, lower right - facing back)

1. Push "RESET"

2. Push "CLEAR"

3. Push "ONLINE" on the keyboard

d. Disc

1. Turn power ON

2. Wait for "LOAD" light

3. Select a disc pack that was not marked "ON" at crash time, and that has a copy of the operating system that one wants. It is possible that the system has crashed while a disc was in the drive or while a disc was not in the drive. The significance is that the boot loader will only get a operting system off of a disc that was not marked "on" when the crash occurred. All discs must be marked "off" before removing so that the system file manager closes all files. A crash or power failure could occur or the user could take his disc out before the disc was marked off. This would make it impossible to load a operating system from that disc until a disc check utility program was executed (DISCHECK). With the exception of the diagnostic disc pack and the disc pack titled "MON 1" (for Monterey number 1) all discs have operating systems on them. They will have an extension "-.001".

4. Insert pack; close cover.

5. Make sure all switches on the Loader Storage Unit are in the OFF position.

6. Make sure there is a cartridge in the drive you are using then switch to "RUN".

e. Computer

1. Turn key to "ON"

2. Make sure "WAIT" is lit, if not press INI".

If "WAIT" does not come back on look at console screen, follow instructions if any. Then go to C.15. If nothing intelligible comes up on the screen attempt to continue at C.3 "DTA".

3. DTA
4. 7
5. 8
6. ADD "78" appears at right in display
7. RD "7A" is displayed on the left on the lights

XXXX
007A = next address

next address contents of location "78"

8. RD "7C" is displayed on the left, one of four codes is displayed on the right, as the contents of location "7A":

C633 Removable upper cartridge
C732 Fixed upper disc
D633 Removable lower cartridge
D732 Fixed lower disc

9. If the code displayed is <u>not</u> the one for the "READY" disc pack with the operating system on it, the code must be changed. For example, if you were going to take the operating system off of the lower removable cartridge:

a. DTA

b. 7

c. A

d. ADD

e. DTA

f. D Code for removable lower cartridge

g. 6

h. 3

i. 3

j. WRT

Check for writing correctly to "7A"

a. DTA

b. 7

c. A

d. ADD

e. RD 7C D633

10. RD 7E B6F1

11. RD location "7E" contains the extension number.

80 0001 (.001)

12. LSU, switch ON

13. Press "INI", listen for the disc to start up

14. Look at Super Bee, "OS32MT03-01"

15. Turn LSU switch OFF again

16. Switch INTERDATA key from ON to LOCK

D. PROCEDURES - POWER DOWN

1. Type DΔM. (Display Map) Note if there are any tasks resident or running.

2. If tasks are running stop them by typing

T	XXXX	where XXXX is the name of the task - (Set task to XXXX)
CAN		cancels the task - and removes from memory if not resident

3. If tasks are not running but resident remove them from memory by typing

T	XXXX	same as in D.2
OPT	NON	(option non-resident)
CAN		cancel and removes from memory

4. Type DΔD (display devices)

5. Mark all discs off. i.e. MAΔD1:, OFF (updates disc directory)

6. Discs

a. Press Run - Load switches to LOAD

b. Wait for LOAD light

c. Take out cartridge

d. Power OFF

7. Super Bee

a. Press RESET

b. Press CLEAR

c. Switch power OFF

8. Printer

a. STOP

b. OFF

9. TEKTRONIX 4014-1 and 4631

a. Power OFF on 4631 (HARDCOPY UNIT)

b. "RESET PAGE" on 4014-1

c. Power OFF (on pedestal)

10. ANN-ARBOR

a. Power OFF logic unit (below right console table)

b. Power OFF on video CRT

11. Computer is usually left on. If it is turned off a cold start will usually be needed to bring it up again. To turn it off, key to OFF.

12. Wall power

a. Phase C, branch 4 main: OFF(center table), Phase A as appropriate.

b. IF Computer key is OFF and AP-12ØB is OFF, then Phase B main OFF. (Usually leave ON.)

"Δ" means one space; "(CR)" means carriage return.

APPENDIX G

CREATING, COMPILING, AND RUNNING
A FORTRAN PROGRAM

1. CREATING A PROGRAM

Programs are written using the utility program EDIT32. This program can be run on many discs by typing the letter "E" or "EDIT32". Before typing to run the EDIT 32 program, the following procedure should be followed:

a. VΔ xxxx to set the system volume, where xxxx is the 4 character mnemonic for the disc pack. Then mark the drive containing that pack ON.

b. Make sure that the system volume has a copy of the Command Substitution program E.CSS or EDIT32.CSS on it, if not, change volumes:

1.	DΔF, - .CSS	Display all files on the system volume with extension .CSS
2.	DΔF, -.CSS, TTY:	Display files of extension type .CSS on the teletype (make sure teletype is ON LINE)
3.	If E.CSS or EDIT32.CSS	does not show up on the list-system volume to another disk pack. That is, suppose the lower cartridge, names MON4:, had a copy on it, then:
	(a) MARK D3:, ON	Physical Location
	(b) VΔMON4:	Volume = MON4

To run the Editor: Read Reference Manual [19].

2. COMPILING A FORTRAN PROGRAM

See "Creating a Program" to store the source (human-readable form) program on the disk. For the purpose of this discussion, assume the FORTRAN program is NEWPROG.FTN.

First make sure the printer is on, use the CSS program. FORT,CSS to compile, assemble, and establish the program as a task.

Example: FORT NEW PROG

(Extension Type .FTN is assumed)

The .CSS file FORT,CSS uses a TET (Task Establisher) file to actually set up the task, and to determine which, if any, libraries should be searched to resolve external references in the source program. If a file of the name NEWPROG.TET exists on the system volume, FORT,CSS will use it to determine the necessary procedure to establish the task. If not, it uses its own standard version TET program, which, among other things, only searches the FORTRAN Runtime libaray to resolve external references. This means that if the user references any other programs (already in object form), one must write a TET file to tell the system where it can find these mysterious programs. (Example: to run the Tektronix Terminal Control System, the user's program references must be resolved against the Tektronix library and the FORTRAN Run Time Libaray.)

Upon successful completion of the FORT.CSS commands, 3 completion codes of zero should be returned to the operator,

one for the FORTRAN Compiler, one for the Assembler and one for the Task establisher. If all of these Completion Codes are zero, and if the printer man shows no undefined terms, the task is ready to be loaded. If the return codes are other than zero, there are errors in the program. The task must be cancelled: CAN, and the listing must be examined to determine where the problems are. Usually the source program will have to be corrected. If undefined terms appear in the printer map, it means that the Task Establisher was unable to find all of the programs and/or subroutines that the user required, in the libraries given in the TET program (either user written or system default). One must ascertain where those programs may be found, and include or edit them in the appropriate TET routine, as described below.

If no subroutines beyond the standard FORTRAN RUN TIME LIBRARY are required, then no further action is required. If, however, AP-12ØB subroutines, PLOT 10 subroutines, etc., then (a .TET file) must be created as follows using the EDIT sequence (or the OS 'BUILD' command. Example follows:

Filename = name. TET where "name" is the same as the "name" in name. FTN

This .TET file should be as follows

(Col.1)

↓

LOG

REMOTE

```
ESTABLISH TASK

MXSPACE 2800

OPTIONS F

GET 600

INCLUDE                         < Includes Main Program and
                                  your own subroutines.

EDIT DAUSUP.OBJ                 < Optional - for DAU support
                                  programs, if required.

EDIT TCSLIB.OBJ                 < Optional - TEK 4014-1 Plot
                                  10 Programs if required.

INCLUDE APSUP.OBJ               < Optional AP-12ØB  programs
EDIT APLIB.OBJ                    if required.

EDIT FVRTL.OBJ                  < Required for all FORTRAN
                                  programs

BUILD TASK, name. TSK           < Here again "name" is the same
                                  as in name. FTN.

MAP

END
```

The "EDIT FVRTL.OBJ" line must immediately precede the "BUILD TASK" line.

3. RUNNING A PROGRAM

If a program has only logical units 5 and 6 (READ(5,-), WRITE (6,-)) required with 5 assigned to CON: (Super Bee KYBD and CRT) and 6 assigned to PR: (line printer), then, assuming the FORT step described previously was successful, proceed to run your program with the RUN.CSS command.

EXAMPLE: RUN NEWPROG

If other devices are required, such as the TEK 4014 or AP-12ØB, then a .ASN assignment file should be built in the

same way as the .TET file using the editor.

This file should be as follows:

```
ASSIGN 1, DAU:      < Required if DAU support routines are
                      used.

ASIGN 7, GDT:       < Required if PLOT 10 routines are used
                      for TEK 4014-1.

ASSIGN 10, APX:     < Required if AP-12ØB routines are used.
ASSIGN 12, APR:
```

(Other assignment statements as required for the READ, WRITE statements in your program. See OS/32MT Reference Manual for use of "Allocate" and "Assign" commands for use with disc files, as required.)

```
$EXIT               < Required.
```

After this file has been built you may proceed with the RUN command as described.

If both the .TET and .ASN files exist, you may combine the FORT and RUN steps with FORTCLG.

EXAMPLE: FORTCLG NEWPROG

4. MISCELLANEOUS OF FORTRAN COMPILATION AND ASSEMBLY

a. This FORTRAN "Compiler" translates source code to objective code. It translates it to assembler. Then the assembler (CAL, for Common, Assembler, Language) translates it to object form.

One must become familiar with the idea that programs use other programs as input. This is how a (real) FORTRAN compiler works. When we write a FORTRAN program, it becomes data for the FORTRAN compiler (which is merely a hugh program,

written in any language) which changes the characters input into other characters, which just happen to be valid machine code, so that if a computer programmer should by some chance try to execute these characters, they would do roughly what the original programmer had intended. When we write the FORTRAN statement, I - 1, the compiler changes it to machine instructions to:

1. Set aside a table of variable references.
2. Insert "I" as the name of a variable.
3. Assign it a unique storage location.
4. Store the value "1" in this location.

The output of this FORTRAN compiler will be a machine program, it can be loaded into memory and executed. TET, CUP, CAL, etc., are all programs that use other programs as input.

b. Definition: Library - A series of objective programs starting with a label, divided by labels, and ending with a label of "ENDVOL". Purpose of a Libaray - To resolve external references of some main program before trying to execute it. EX - Main routine calls SUBROUTINE A - A calls B, B calls C. These routines are order ABC in the library. If ordered ACB, then A would link to our main routine. But since C was not called by A it will not be linked. A called B, link it now, at the end of the file, and B needs C, but there is no C.

c. Always delete the old .TSK file prior to a new FORT step.

d. Disc space is limited, please delete old files not in use.

e. Local CRT Driver was modified December 1977 to change control characters. Character '#' (X'23') and "_" (underscore)(X'5F') are now valid printing characters. The function of line delete is provided by 'control X' (X'08') and the character delete (backspace) by 'Control H' (X'18'). This change was required to permit the full range of Graphics Input characters of the TEKTRONIX 4014-1.

Devices affected:

GDT: TEKTRONIX 4014-1

CON: Super Bee KYBD/Video Display

For TTY: ' ' is character delete

'#' is line delete

f. When writing an interactive program, that is, a program that actually waits for the user to supply data, via the Console's Teletype, etc., be sure to include a write preceding every read, to tell the user what is expected.

Example:

```
Write (6, 10)
10 Format ('enter frequency in Hz, Format
           F8.1')
Read (5, 20) Frequency
20 Format (F8.1)
```

Without this write, the computer merely stops, waiting for input of "Freq." without telling the person at the console what it is waiting for.

APPENDIX H

LOADING INTERDATA PROVIDED DIAGNOSTIC

These diagnostics are on paper tape, disc and magnetic tape. The primary references for these diagnostics are the Multi-Media Diagnostic Loader (32 Bit), Publication I/B06-176FØ2, Interdata, Inc., April 1975, and the individual test program manual. This system has the ability to load diagnostics from paper tape, the magnetic tape and the disc. Loading via paper tape will not be covered. Loading from the disc is faster than loading from the magnetic tape. The sequence follows.

1. Loading Diagnostic Programs From Disc

    a. Place Diagnostic Magnetic Tape on the drive.

    b. Power it up and load the tape.

    c. Place diagnostic disc in upper drive.

    d. On the hexadecimal display, enter the following:

```
DTA
78
ADR
DTA
  (95A1
  (45A1     if the left cassette drive is being used)
  (55A1     if the right cassette drive is being used)
WRT
C600        (D600 if disc is going to be in lower drive)
WRT
0001
WRT
DTA
30
DTA
WRT
DTA
WRT         to zero 30, 31
```

```
DTA
WRT         to zero 32, 33
DTA
WRT         to zero 34, 35
DTA
5Ø
WRT
DTA
5Ø
ADR
DAT
D5ØØ
WRT
DTA
ØØCF
WRT DTA
43ØØ
WRT
DTA
ØØ8Ø
WRT
DTA
3Ø
ADR
DTA
RUN
```

e. Observe "FFFF" in the right most section of the display. If this is not present, retry the process again. If it continues to fail, go to the primary reference and get help!

f. If you had a good load "FFFF" will be present. This means you may now use this program to load the diagnostics on the disc pack labeled diagnostic.

To restart this program, <u>which must be done every time you want a different diagnostic</u>, first attempt the following:

```
DTA
6ØØØ
ADR
RUN
```

If "FFFF" comes up again you are in luck. If it does not, reload the Multi-Media loader by going to C.

g. To load the diagnostic push:

```
DTA
XXXX
RUN
```

Where XXXX is the sequence number to the diagnostic you desire to run as shown in Table IX.

2. Loading Diagnostics From Magnetic Tape

a. Place magnetic tape on the drive.

b. Power drive up and load the tape.

c. On the hexadecimal display enter the following:

```
DTA
3Ø
ADR
DTA
WRT       to zero 3Ø, 31
DTA
WRT       to zero 34, 35
DTA
5Ø        to load address 36 with a "5Ø"
WRT
DTA
5Ø
ADR
DTA
D5ØØ
WRT
DTA
ØØCF
WRT
DTA
43ØØ
WRT
DTA
Ø8ØØ
WRT
DTA
78
ADR
DTA
```

```
95A1
WRT
DTA
WRT
DTA
WRT
DTA
1
WRT
DTA
3Ø
ADR
DTA
RUN
```

d. Same as 1.d

e. Same as 1.e except now you are going to get the diagnostics off of the tape.

f. Same as 1.d

g. Same as 1.d

TABLE IX

DIAGNOSTIC LISTINGS

SEQ #	06- #	REV.	NAME	LOW	HIGH
001	176	00	F01 MULTI MEDIA DI. LDR. 16BIT	04000	04A1F
002	176	00	F02 MULTI MEDIA DI. LDR. 32BIT	06000	06B1F
003	177	00	F01 MULTI MEDIA DI. GEN. 16BIT	05000	06F47
004	177	00	F02 MULTI MEDIA DI. GEN. 32BIT	07000	09069
010	003	08	MEMORY TEST PART 1	01000	0152F
011	003	08	MEMORY TEST PART 2	00080	0101B
012	003	08	MEMORY TEST PART 3	00080	004CD
013	106	06	PROCESSOR TEST PART 1	002D0	01FFF
014	106	06	PROCESSOR TEST PART 2	002D0	01D62
015	128	01	MODEL 50 PROCESSOR TEST	00080	01FA0
016	135	01	MEMORY PROTECT TEST	00080	00BFA
017	143	00	MOS MEMORY HOLD TEST	00080	00305
018	144	00	MOS PARITY INITIALIZE TEST	00080	000A3
019	158	00	SERIES 32 BASIC TEST	00080	0029F
01A	153	00	MODEL 7/32 HW PROCESSOR TEST	00A00	02C18
01B	154	02	SERIES 32 PROCESSOR TEST PART1	00A00	03F1C
01C	155	01	SERIES 32 PROCESSOR TEST PART2	00A00	0307B
01D	156	02	F01 SER. 32 MEMORY TEST PART 1	02000	02663
01E	156	02	F02 SER. 32 MEMORY TEST PART 2	00A00	0184F
01F	156	02	F03 SER. 32 MEMORY TEST PART 3	00A00	014CB
020	148	00	SERIES 6A MEMORY TEST PART1	00080	004AB
021	148	00	SERIES 6A MEMORY TEST PART2	01000	012EB
022	159	02	SERIES 32 SYSTEM EXERCISER	00A00	059D7
024	162	00	02-340 MEMORY TEST PART 1	00080	003C3
025	162	00	02-340 MEMORY TEST PART 2	01000	01345
026	178	00	SERIES 32 PROCESSOR TEST PART3	00A00	01A7B
027	157	00	32 BIT SER. 6A MEM. TEST PART1	04000	0484F
028	157	00	32 BIT SER. 6A MEM. TEST PART2	00A00	01617
102	071	00	801 AUTO CALL UNIT TEST	00080	002FD
103	101	00	DIGITAL MULTIPLEXOR TEST	00A00	00FBF
106	127	01	PALS OFF-LINE TEST	00A00	02B7B
109	132	02	201/301 DATA SET ADAPTER TEST	00400	01C27
10E	140	00	DYNAMIC CONTROL STORAGE TEST	00A00	018B5
110	147	00	CONVERSION EQUIPMENT TEST	00A00	0233B
111	149	00	F01 MUX BUS SWITCH TEST PART1	00A00	01001
112	149	00	F02 MUX BUS SWITCH TEST PART2	00A00	00E61
113	150	00	SENSE CONTACT MODULE TEST	00A00	00F5F
114	151	00	RELAY DRIVER TEST	00A00	00E53
115	163	01	GRAP. DISPLAY TERMINAL TEST	00A00	017CB
117	166	00	F01 SA 360/370 IF. TEST INTER.	00A00	02E7B
118	167	00	F01 MA 360/370 IF. TEST INTER.	00A00	03955
119	179	00	16 BIT SELCH TEST	00A00	01ADD
120	165	03	SERIES 32 LSU SUPPORT PROGRAM	00A00	0279E
200	161	01	EXTENDED SELECTOR CHANNEL TEST	00A00	01B11
202	160	02	F01 MEMORY ACCESS TEST PART 1	00A00	01C1B
203	160	02	F02 MEMORY ACCESS TEST PART 2	0FFF8	10D6D
300	168	00	COMMON HSPT READER/PUNCH TEST	00A00	01F0F
301	169	00	COMMON CARD READER TEST	00A00	0211D
302	170	01	COMMON LINE PRINTER TEST	00A00	01BF7
303	171	00	COMMON CASSETTE TEST	00A00	032E5
304	172	00	COMMON MAG TAPE TEST	00A00	02FC9
305	173	00	F01 COMMON DISC TEST	00A00	036B1
306	173	00	F02 COMMON DISC FORMATTER	00A00	02B0F

TABLE IX
(con't)
DIAGNOSTIC LISTINGS

SEQ #	06- #	REV.	NAME	LOW	HIGH
307	004	08	COMMON TELETYPE TEST	00A00	01ADF
308	129	06	COMMON UNV. LOGIC INTER. TEST	00A00	00E1B
309	133	03	COMMON UNV. CLOCK MOD. TEST	00A00	01E66
310	134	02	COMMON 8 LINE INTERRUPT TEST	00A00	01E2A
311	146	03	COMMON CRT TEST PROGRAM	00A00	01A43
312	925	00	32BIT DISC DIAG/FORMT	00A00	01428
21B	154	01	SERIES 32 PROCESSOR TEST PART1	00A00	03F3F
400	172	00	MAG TAPE TEST (CRT)	00A00	04000
401	173	00	WANG DISC TEST PROGRAM (CRT)	00A00	04000
402	173	00	WANG DISC FORMATTER PROG (CRT)	00A00	04000
403	168	00	HSPTR/P TEST (CRT)	00A00	03000
404	127	04	PALS TEST (CRT)	00A00	04000
405	165	03	LSU SUPPORT PROGRAM (CRT)	00A00	04000
313	067	00	32BIT REL LOADER	1FB00	1FFFF
314	062	03	32BIT XAIDS - LO CORE	05000	067EC
315	062	03	32BIT XAIDS - HI CORE	1E300	1FAEC
500	132	03	COMMON 201/301 DSA TEST	00A00	03088
501	127	04	COMMON PALS-OFF LINE TEST	00A00	03750
503	182	00	MEMORY ACCESS MULTIPLEXOR(MAM)	00A00	04940
600	999	05	TELEFILE DISC TEST	00400	04C2C

APPENDIX I

SERIES 32 BASIC TEST

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 019, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, 0080, ADR, RUN -the program is now running

Observe F2 in display and follow page A2-1 in BASIC TEST MANUAL. Program MUST stop after "S32BT". If it does not start again at 0080. The alphabet and numbers 0-9 will follow after a second push on the run button.

You may then input from the keyboard to check the echo.

Sample Output:

```
S32BT
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

INPUT FROM KEY BOARD.....
1234567890:-QWERTYUIOP
ASDFGHJKL;ZXCVBNM,./
```

APPENDIX J

SERIES 32 PROCESSOR TEST PART 1

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 1B, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Type 7X on TTY and observe display count to A for test numbers 2-A.

Sample Output:

```
S32PT1 R02
CPU
*
7X
NO ERROR

000A  0000
```

APPENDIX K

SERIES 32 PROCESSOR TEST PART 2

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 1C, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Type 7X on teletype after teletype stops printing and follow instructions. After subtest enter the number for the test you desire. We have tests 1, 2, 3, 5, 6, 7. After entering a number hit return key.

The "BRK" stands for BREAK key on teletype -

Function 0 the FN key and 0 key on the hexadecimal display.

Power Off/On is the key switch on the hexadecimal display -

On Test 6 observe the hexadecimal display step through each part of test 6 by pushing the break key. After F is printed press break key and return key.

Sample Output:

```
S32PT2 R01
CPU
*
7X
SUBTEST
*
```

Sample Output (Continued)

```
1
DEPRESS KEYS
1234567890
1234567890
DEPRESS KEYS
1234567890
1234567890
NO ERROR

SUBTEST
*
2
PRESS BRK
NO ERROR

SUBTEST
*
3
PRESS BRK
FUNCTION 0
NO ERROR

SUBTEST
*
5
1234567890
1234567890
ABCDEFGHIJ
1234567890
DEPRESS KEYS
1234567890
1234567890
1234567890
DEPRESS KEYS
1234567890
1234567890
ABCDEFGHIJ

SUBTEST
*
6
ABCDEF
```

Sample Output (Continued)

```
SUBTEST
*
7
INITIALIZE
PRESS BRK
POWER OFF/ON
PRESS BRK
NO ERROR
```

APPENDIX L

SERIES 32 PROCESSOR TEST PART 3

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 261C, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Type 7X on teletype and follow instructions.

Function Ø one FN and 0 key on hex decimal display

Sample Output:

```
S32PT3 R00
CPU
*
7X
MAC RESPONSE AT   000300

SUBTEST
*
1
NO ERROR

SUBTEST
*
2
NO ERROR
SUBTEST
*
#
FUNCTION 0
FUNCTION 0
FUNCTION 0
```

Sample Output (Continued)

```
DEPRESS KEYS
1234567890
1234567890
DEPRESS KEYS
1234567890
1234567890
NO ERROR

SUBTEST
*
```

APPENDIX M

SERIES 32 MEMORY TEST PART 1

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 1D, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, 2000, ADR, RUN -the program is now running

Press Run again - observe an "*"

Press Return Key and observe the sample output on the teletype.

```
S32MT1 06-156F01R02
*

01
02
03
04
05
06
NO ERROR
*
```

APPENDIX N

SERIES 32 MEMORY TEST PART 2

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 1E, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Observe on teletype the available. One board of memory represents 32k words - for 8 boards that is 262. Available memory is printed out from the lowest location to the highest location. Example: 0000 - 3FFFF would mean that boards .1 through 8 are good. If a board or part of a board is bad on would observe the available memory broken up - Example:

00000 - 17FFF
2000 - 2BFFF
2E000 - 2FFFF

The Board Breakdown by memory is as follows:

MEMORY BOARD NUMBER	MEMORY
1	0 - 7FFF
2	8000 - FFFF
3	10000 - 17FFF
4	18000 - 1FFFF
5	20000 - 27FFF
6	28000 - 2FFFF
7	30000 - 37FFF
8	38000 - 3FFFF

For one run, type 0, Return

For continuous Runs, Type L, Return. The BRK Key will terminate continuous runs.

Sample Output:

```
S32MT2 06-156F02R02
AVAILABLE MEMORY
00000 -3FFFF

SUBTEST
* 1
01
NO ERROR

SUBTEST
* 2
02
NO ERROR

SUBTEST
* 3
03
NO ERROR

SUBTEST
* 4
04
NO ERROR

SUBTEST
* 5
05
NO ERROR

SUBTEST
* 6
06
NO ERROR

SUBTEST
* 7
07
NO ERROR
```

Sample Output (Continued)

```
SUBTEST
* 0
01
NO ERROR
02
NO ERROR
03
NO ERROR
04
NO ERROR
05
NO ERROR
06
NO ERROR
07
NO ERROR

SUBTEST
*
```

APPENDIX O

SERIES 32 MEMORY TEST PART 3

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 1F, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Observe *, hit carriage return.

Observe the hexadecimal display count through memory -

For continuous test type L and return -

Sample Output:

```
S32MT3 06-156F03R02
AVAILABLE MEMORY
00000- 3FFFF

*

NO ERROR

*
```

APPENDIX P

MEMORY ACCESS TEST PART 1

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 202, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Type Run and observe results -

Sample Output:

```
MACT 06-160F01R02
AVAILABLE MEMORY
00000- 3FFFF

* RUN
TEST  00
NO ERROR
TEST 01
NO ERROR
TEST 02
NO ERROR
TEST 03
NO ERROR
TEST 04
NO ERROR
TEST 05
NO ERROR
TEST 06
NO ERROR
```

Sample Output (Continued)

```
TEST 07
NO ERROR
TEST 08
NO ERROR
TEST 09
NO ERROR
TEST 0A
NO ERROR
*
```

APPENDIX Q

MEMORY ACCESS TEST PART 2

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 203, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, 10000, ADR, RUN -the program is now running

Type Run and observe results -

Sample Output:

```
MACT 06-160F02R02
* RUN
TEST 00
NO ERROR
TEST 01
NO ERROR
TEST 02
NO ERROR
TEST 03
NO ERROR
TEST 04
NO ERROR
TEST 05
NO ERROR
TEST 06
NO ERROR
TEST 07
NO ERROR
*
```

APPENDIX R

COMMON LINE PRINTER

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 302, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Type Run on teletype -

Observe results on teletype and printer - Be sure printer in on!!

Sample Output from Teletype:

```
COMMON LINE PRINTER TEST 06-170R01

*RUN

TEST 00
NO ERROR
TEST 01
NO ERROR
TEST 02
NO ERROR
TEST 03
NO ERROR
END OF TEST

*
```

APPENDIX S

COMMON CASSETTE TEST

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 303, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Be sure the tape heads are clean

As you place the cassette in the drive have the take-up on your left and supply on your right.

Type "DEVADR 45" or "55" for testing the device - 45 for left drive, 55 for right drive.

Type "RUN".

Observe the tape drive.

There are options-to see what they are type option.

Sample Output:

```
COMMON CASSETTE TEST PROGRAM 06-171R00
*DEVADR 45
*RUN
TEST 00
NO ERROR
TEST 01
NO ERROR
TEST 02
NO ERROR
TEST 03
NO ERROR
```

Sample Output (Continued)

```
TEST 04
NO ERROR
TEST 05
NO ERROR
TEST 06
NO ERROR
END OF TEST
```

APPENDIX T

COMMON MAG TAPE

It is assumed that the multi-media loader has been loaded as described in Appendix H, and "FFFF" is in the right-most position on the hexadecimal panel.

On the hexadecimal panel push:

DTA, 304, RUN -wait for "FF" to appear

Once the "FF" appears then enter:

DTA, A00, ADR, RUN -the program is now running

Type "DEVADR 95"

Type Run

There are options - Type "OPTION" to list them.

Observe tape drive.

Sample Output:

```
COMMON MAGNETIC TAPE TEST PROGRAM 06-172R00

*OPTION
TEST  0,1,2,3,4,5
LOOP   0000
CONTIN 0000
NOMSG  0000
DEVADR 0085
DV2ADR 0000
SELCH  00F0
INTLEV 0000
DEVICE 0000
MODE   0000
TRACK  0009
RECFIL 0100
BYTES  00FF
FILES  0001
REPEAT 0003
```

Sample Output (Continued)

```
IRG     0010
DU      0000
READ    0001
WRITE   0001
BKSPAC  0001
SKIP    0001
WEOF    0000
COMPAR  0001
CRC     0000
RDCRC   0000
DUMP    0000
DATA    0001
SCOPE   0000
TIME    0800
*DEVADR 95
*RUN

TEST 00
NO ERROR
TEST 01
NO ERROR
TEST 02
NO ERROR
TEST 03
NO ERROR
TEST 04
NO ERROR
TEST 05
NO ERROR
END OF TEST

*
```

APPENDIX U

On the following pages are the program that are from the SATCOM Spectrum Analysis System. They are in order:

MAIN
LINK
BLOCK DATA
SPECTRL (C4)
ILLEGAL ENTRY HANDLER
WAIT MS
TEKTRO

Following the above programs are other programs. One is a FORTRAN program; the others are CSS programs.

```
      COMMON/BLK1/LGTAB(78,10),IDEV,NUM,DECMPT,NUMBER(16)
     1,NUMREG,FNUMRG,PT,POWER, PONTR
     *,JFREQ, REMBER,INDEX,IIFREQ,LPFBW,IFATN
     *,IFREQ(24,4)
      COMMON/BLK3/CMD,HOLD,STACK(10),STATE,MODE,BUTLIS(18,6)
      COMMON/BLK4/F(1025),I2DAT1(2050),I2DAT2(2050)
     1,BUFFER(33),DAY(2),HOUR(2),MONT(12)
     *,VMAX,VMIN,IX1,IX2,IY1,IY2,XMIN,XMAX,YMIN,YMAX,ILOC,
     *IPUT
      LOGICAL DECMPT
      ADDRESS INTRPT
      INTEGER*4 PT,POWER,STACK,PONTR,STATE,LGTAB
     *,IDEV,NUM,II,JJ,NUMBER,ICNT,IADC,ITIME,IAVG,IPLOT,
     *,ILNLG,JFREQ,MODE,ICHN,REMBER,INDEX,IIFREQ,IFATN
     *,CMD,HOLD,BUTLIS
      REAL F(1025),LPFBW
      INTEGER*2 I2DAT1(2050),I2DAT2(2050)
      INTEGER BUFFER(33),DAY(2),HOUR(2),MONT(12)
      CALL LINK
      DO 30 I=1,2050
      I2DAT1(I)=1
      I2DAT2(I)=2
30    CONTINUE
      END
```

```
      SUBROUTINE LINK
      COMMON/BLK1/LGTAB(78,10),IDEV,NUM,DECMPT,NUMBER(16)
     1,NUMREG,FNUMRG,PT,POWER, PONTR
     *,JFREQ, REMBER,INDEX,IIFREQ,LPFBW,IFATN
     *,IFREQ(24,4)
      COMMON/BLK3/CMD,HOLD,STACK(10),STATE,MODE,BUTLIS(18,6)
      LOGICAL DECMPT
      ADDRESS INTRPT
      INTEGER*4 PT,POWER,STACK,PONTR,STATE,LGTAB
     *,IDEV,NUM,II,JJ,NUMBER,ICNT,IADC,ITIME,IAVG,IPLOT,
     *,ILNLG,JFREQ,MODE,ICHN,REMBER,INDEX,IIFREQ,IFATN
     *,CMD,HOLD,BUTLIS
      REAL F(1025),LPFBW
      INTEGER*2 I2DAT1(2050),I2DAT2(2050)
      INTEGER BUFFER(33),DAY(2),HOUR(2),MONT(12)
      DATA MONT/'JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV
     * DEC '/
C        CLEAR DEVICES A25,A26,C4,TEKTRONICS 4014
1     CALL APCLR
      DO 100 I=1,78
      CALL KTLWR(4,-I)
100   CONTINUE
      CALL TEKTRO
C=====================================================================
C            SET UP FOR INITIAL RUN
C=====================================================================
      ICNT=256
      IADC=1
      ITIME=1
      IAVG=10
      IPLOT=1
      ILNLG=?
      FREQ=10000.0
      VMAX=0.0
      VMIN=0.0
      IPLOT=1
      ILNLG=2
      MODE=1
      STATE=4
      PONTR=1
      INTRPT=A'1111'
      CNT=1
      CMD=3
      MS=00001
      HOLD=0
C WRITE TO STATUS  SOME MESSAGES
      WRITE (4,401)
      CALL WAIT(MS)
      WRITE(4,402)
401   FORMAT('INSTRUCTION:',/
     1'I WILL TALK TO YOU BY WRITING',/
     1'HERE AND BY LIGHTING UP C4',/
```

```
     1'THE INFORMATION WILL BE ',/
     1'REMINDERS;WHAT I AM DOING',/
     1'THE NUMBER AS YOU PUSH THE',/
     1'NUMBER PAD AND ANY PROMPTS',/
     1'GOOD-LUCK AND LETS GO !!!',/)
402   FORMAT('HERE IS THE PRESENT STATUS: ',/
     1/////////////)
       CALL SETUP(ICNT,FREQ,IADC,ITIME,IAVG,ILNLG,IPLOT,
     *SCALE,ICHN)
C=============================================================
C      CHECK FOR COMMANDS
C=============================================================
       CALL KTLON(4,INTRPT)
719   GO TO (720,730,770,750),CMD
720   IF(HOLD.EQ.0)GO TO 730
      HOLD=0
      J=BUTLIS(1,MODE)+1
      DO 723 I=2,J
      CALL KTLSIM(4,BUTLIS(I,MODE))
723   CONTINUE
730   CMD=3
      CALL SPECTR(ICNT,IADC,ITIME,IAVG,IPLOT,ILNLG,ICHN,
     *FREQ,VMAX,VMIN)
770   GO TO 719
750   GO TO 1
 1111 CALL KTLRD(IDEV,NUM)
       CALL C4(ICNT,IADC,ITIME,IAVG,IPLOT,ILNLG,ICHN,FREQ)
      RETURN
      END
```

```
 BLOCK DATA
 COMMON/BLK1/LGTAB(78,10),IDEV,NUM,DECMPT,NUMBER(16)
1,NUMREG,FNUMRG,PT,POWER, PONTR
*,JFREQ, REMBER,INDEX,IIFREQ,LPFBW,IFATN
*,IFREQ(24,4)
 INTEGER*4 PT,POWER,STACK,PONTR,STATE,LGTAB
*,IDEV,NUM,NUMBER,ICNT,IADC,ITIME,IAVG,IPLOT,ILNLG
*,JFREQ,MODE,ICHN,REMBER,INDEX,IIFREQ,IFATN
*,CMD,HOLD,BUTLIS
*,IFREQ(24,4)
 REAL F(1025),LPFBW
 INTEGER*2 I2DAT1(2050),I2DAT2(2050)
 INTEGER BUFFER(33),DAY(2),HOUR(2),MONT(12)
 DATA PT,POWER,NUMREG/3*0/
 DATA NUMBER/7,8,9,0,0,4,5,6,0,1,2,3,4*0/
 DATA LGTAB/1,5*0,1,23*0,1,0,1,2*0,1,4*0,1,30*0,1,6*0
1,48*0,17*1,0,1,11*0,
11,1,4*0,4*1,0,0,12*1,2*0,3*1,0,1,0,1,3*0,5*1,0,3*1,3*0
*,17*0,0,1,
*0,3*1,2*1,1,1,3*0,
*2*1,4*0,4*1,61*0,1,0,2*1,3*0,
1468*0/
 DATA IFREQ /988500000,988750000,989000000,
C989250000,989500000,989750000,990000000,990250000,
C990500000,990750000,991000000,991250000,991500000,
C991750000,992000000,992250000,992500000,992750000,
C993000000,993250000,1041500000,1075500000,
C0,0,1004500000,1019500000,1036500000,1053500000,
C1069500000,1084500000,1153500000,1168500000,1182500000,
C1197500000,939450000,939550000,939600000,939650000,
C939700000,939750000,939800000,939850000,939900000,
C939950000,940000000,940100000,1106000000,0,
C1005500000,1020500000,1037500000,1054500000,1070500000,
C1085500000,1154500000,1169500000,1183500000,1198500000,
C940450000,940550000,940600000,940650000,940700000,
C940750000,940800000,940850000,940900000,940950000,
C941000000,941100000,1117000000,0,
C1006500000,1021500000,1038500000,1055500000,
C1071500000,1086500000,1155500000,1170500000,1184500000,
C1199500000,941450000,941550000,941600000,941650000,
C941700000,941750000,941800000,941850000,941900000,
C941950000,942000000,942100000,1123000000,0/
 END
```

```
      SUBROUTINE C4(ICNT,IADC,ITIME,IAVG,IPLOT,ILNLG,ICHN,
     *FREQ)
      COMMON/BLK1/LGTAB(78,10),IDEV,NUM,DECMPT,NUMBER(16)
     1,NUMREG,FNUMRG,PT,POWER, PONTR
     *,JFREQ, REMBER,INDEX,IIFREQ,LPFBW,IFATN
     *,IFREQ(24,4)
      COMMON/BLK3/CMD,HOLD,STACK(10),STATE,MODE,BUTLIS(18,6)
      LOGICAL DECMPT
      INTEGER*4 PT,POWER,STACK,PONTR,STATE,LGTAB
     *,IDEV,NUM,NUMBER,ICNT,IADC,ITIME,IAVG,IPLOT,ILNLG
     *,JFREQ,MODE,ICHN,REMBER,INDEX,IIFREQ,IFATN
     *,CMD,HOLD,BUTLIS
      REAL F(1024),LPFBW
C             TURN OFF ENTRY REQUIRED
       CALL KTLWR(4,-63)
C TEST IF THIS WAS LEGAL; WAS ANYTHING SET TO SAY WHAT IS
C NOT LEGAL?
C IF THE NUMBER IS ALLOWED THAN IT IS LEGAL AND TRUE OTHER-
C WISE FALSE
      IF (LGTAB(NUM,STATE).EQ.0) CALL ILEHDL
C NOW CHECK IF THE NUMBER PAD IS REQUIRED; IF SO THEN THEN
C CHANGE THE
C LEGAL ENTRY TABLE. IF NOT CONTINUE
      IF((NUM.GT.48).OR.((NUM.LT.37).AND.(NUM.GT.9))
     1.OR.(NUM.LT.8))GO TO 200
100   CONTINUE
C HERE WE WILL CHANGE THE LGTAB ENTRY;SOME HOW KEEPING THE
C OLD
      STACK(PONTR)=STATE
      PONTR=PONTR+1
      STATE=2
200   CONTINUE
CNOW CHECK FOR THE VARIOUS NUM AND PERFORM THEIR SPECIFIC
C ACTIONS
      GO TO (1000,1000,1000,1000,9999,9999,
     *7000,8000,9000,10000,11000,9999,
     *13000,14000,15000,16000,17000,9999,
     *19000,19000,19000,19000,19000,19000,
     *9999,9999,27000,28000,29000,9999,
     *31000,9999,33000,9999,9999,9999,
     *37000,38000,39000,40000,41000,9999,
     *43000,44000,45000,9999,9999,9999,
     *49000,49000,49000,52000,53000,49000,49000,49000,
     *57000,49000,49000,49000,61000,9999,9999,49000,
     *65000,9999,67000,9999,69000,70000,71000,
     *72000,72000,74000,72000,72000,9999),NUM
C                                          FREQ PLAN ROW
1000  JFREQ=NUM
      GOTO 20
C                                     MODE ROW
7000  MODE=1
      STATE=4
```

```
       FREQ=1000000.0
      ICNT=256
      IAVG=20
      IPLOT=1
      ICHN=21
      SCALE=10.0
      ITIME=0
      IADC=2
      ILNLG=2
C SET UP RECEIVER RCVR CHN=C;ATN=0DB; IF FLTR=1MHZ;
C L.O.=99.1MHZ
      ISTAT= KTLWR(5,Y'100')
      ISTAT=KTLWR(5,Y'37F')
      ISTAT=KTLWR(5,Y'218')
      CALL KTLWR(7,991000000)
      CALL KTLWR(4,15)
      CALL KTLWR(4,21)
      CALL KTLWR(4,29)
      CALL KTLWR(4,33)
      GO TO 20
3000   MODE=2
      REMBER=NUM+39
       FREQ=20000.0
      ICNT=256
      IAVG=20
      IPLOT=1
      SCALE=10.0
      ITIME=0
      IADC=1
      ILNLG=2
C SET UP THE RECEIVER RCVR CHN=C; IF FILTR=30KHZ; ATN =0DB
       CALL KTLWR(5,Y'37F')
       ISTAT=KTLWR(5,Y'230')
       ISTAT=KTLWR(5,Y'100')
       CALL KTLWR(4,15)
      CALL KTLWR(4,21)
      CALL KTLWR(4,28)
      CALL KTLWR(4,33)
      GO TO 20
9000  MODE=NUMREG+2
      REMBER=NUM+37
      GO TO 20
10000 STATE=3
      GO TO 20
11000 ICNT=256
       FREQ=1000000.0
      IAVG=20
      IPLOT=20
      ICHN=11
      SCALE=10.0
      ITIME=0
      IADC=2
```

```
      ILNLG=2
C SET UP THE RECEIVER
       ISTAT=KTLWR(5,Y'1C0')
       ISTAT=KTLWR(5,Y'37F')
       ISTAT=KTLWR(5,Y'230')
C SET THE LOCAL OSC UP
       CALL KTLWR(7,1041500000)
       CALL KTLWR(9,1075000000)
       CALL KTLWR(4,15)
      CALL KTLWR(4,21)
      CALL KTLWR(4,29)
      CALL KTLWR(4,33)
C                                          IF FILTER ROW
13000 ISTAT= KTLWR(5,Y'220')
       GO TO 20
14000 ISTAT= KTLWR(5,Y'228')
       GO TO 20
15000 ISTAT= KTLWR(5,Y'230')
         GO TO 20
16000 ISTAT=KTLWR(5,Y'238')
           GO TO 20
17000 ISTAT=KTLWR(5,Y'218')
       GO TO 20
C                                        FFT BLOCK SIZE ROW
19000 ICNT=2**(NUM-19)*64
      GO TO 20
C                                         OPTIONS ROW
27000 IF (ITIME.EQ.1)GO TO  27001
      ITIME=1
       CALL KTLWR(4,27)
      GO TO 27002
27001 ITIME=0
      CALL KTLWR(4,-27)
27002 WRITE(4,27003) ITIME
27003 FORMAT(' ITIME= ',I1)
      GO TO 20
28000 IADC=1
      CALL KTLWR(4,-29)
      GO TO 20
29000 IADC=2
      CALL KTLWR(4,-28)
      GO TO 20
C                              PLOT ROW
31000 ILNLG=1
      GO TO 20
33000 ILNLG=2
      GO TO 20
C                                    DATA ENTRY ROW
37000 CONTINUE
      MODE=10
      REMBER=NUM
      WRITE(4,37040)
```

```
      WRITE(4,37041)
37040 FORMAT(' ENTER CHANNEL NUMBER')
37041 FORMAT(' THEN PRESS ENTER ',/' OR CLEAR ENTRY')
      GO TO 20
38000 CONTINUE
      REMBER=NUM
      WRITE(4,37042)
      WRITE(4,37041)
37042 FORMAT(' ENTER CENTER FREQUENCY')
      GO TO 20
39000 CONTINUE
      REMBER=NUM
      WRITE(4,37043)
      WRITE(4,37041)
37043 FORMAT(' ENTER LP FILTER B.W.',/'1MHZ,100,30,10,3 KHZ')
      GO TO 20
40000 CONTINUE
      REMBER=NUM
      WRITE(4,37044)
      WRITE(4,37041)
37044 FORMAT(' ENTER A TO D RATE')
      GO TO 20
41000 REMBER=NUM
      WRITE(4,37048)
37048 FORMAT('ENTER THE NUMBER OF PLOTS DESIRED')
      WRITE(4,37041)
      GO TO 20
43000 CONTINUE
      REMBER=NUM
      WRITE(4,37045)
      WRITE(4,37041)
37045 FORMAT(' ENTER IF ATTENUATION')
      GO TO 20
44000 GO TO 9999
      REMBER=NUM
      WRITE(4,37046)
      WRITE(4,37041)
37046 FORMAT(' ENTER BANDWIDTH DESIRED FOR PLOT')
      GO TO 20
45000 REMBER=NUM
      WRITE(4,37047)
      WRITE(4,37041)
37047 FORMAT(' ENTER NUMBER OF BLOCKS TO BE AVG')
      GO TO 20
C                                     NUMBERS
49000 CONTINUE
      IF (DECMPT) PT=PT+1
C     BY THIS TIME WHAT EVER HAS GOTTEN THRU SHOULD ONLY BE
C A NUMBER
C     NOW DO THE TRANSLATION
      NUMREG=NUMREG*10+NUMBER(NUM-48)
      B=NUMREG*10**(POWER-PT)
```

```
      WRITE (4,48)B
48     FORMAT(G20.6)
      GOTO 20
52000 POWER=6
      GO TO 20
C                                     ENTER
53000 CONTINUE
50    A=NUMREG
       W=FLOAT(POWER-PT)
       FNUMRG=A*10.0**(W)
       NUMREG=FNUMRG
C NOW TURN THE NUMBER FLAG OFF
      PT=0
      POWER=0
      INDEX=REMBER-36
      PONTR=PONTR-1
      STATE=STACK(PONTR)
      GO TO (3700,3800,3900,4000,4100,9999,4300,4400,4500,
     *4600,4700),INDEX
3700  ICHN=NUMREG
      NUMREG=0
      PT=0
      POWER=0
      MODE=10
      CALL KTLWR(7,IFREQ(ICHN,JFREQ))
      WRITE(4,3840)ICHN
3840  FORMAT('CHANNEL',I2)
      GO TO 20
3800  C=FNUMRG-150000000
      IIFREQ=C*10
      NUMREG=0
      PT=0
      POWER=0
      CALL KTLWR(7,IIFREQ)
      WRITE(4,3850)FNUMRG
3850  FORMAT('CENTER FREQ AT ',G12.5)
      GO TO 20
3900  LPFBW=FNUMRG
      NUMREG=0
      PT=0
      POWER=0
      CALL KTLWR(13,LPFBW)
      WRITE (4,3940)LPFBW
3940  FORMAT('LP FILTER BW IS ',G12.5)
      GO TO 20
4000  FREQ=FNUMRG
      NUMREG=0
      PT=0
      POWER=0
      WRITE(4,4040)FREQ
4040  FORMAT('SAMPLE RATE OF ',G12.5)
      GO TO 20
```

```
4100  IPLOT=NUMREG
      NUMREG=0
      PT=0
      POWER=0
      WRITE(4,4140)IPLOT
4140  FORMAT('THE NUMBER OF 3-D PLOTS',/'WILL BE ',I2)
      GO TO 20
4300  IFATN=NUMREG
      NUMREG=0
      PT=0
      POWER=0
      IHEX=X'37F'-IFATN
      ISTAT=KTLWR(5,Y'37F'-IFATN)
      WRITE(4,4340)IFATN
4340  FORMAT('IF ATTENUATIONIS ',I3,' DB')
      GO TO 20
4400  GO TO 9999
4500  IAVG=NUMREG
      NUMREG=0
      PT=0
      POWER=0
      WRITE(4,4540)IAVG
4540  FORMAT(I6,' BLOCKS ARE TO BE AVERAGED')
      GO TO 20
4600  CONTINUE
      NUMREG=0
      PT=0
      POWER=0
      STATE=4
C STUFF IN HERE THE AUTO NUMBER JUNK
      GO TO 9999
      GO TO 20
4700  ICHN=NUMREG
      NUMREG=0
      PT=0
      POWER=0
      STATE=4
      CALL KTLWR(7,IFREQ(ICHN,JFREQ))
      GO TO 20
C                              POWERS, DECIMAL PT, CLEAR ENTRY
57000 POWER=3
      GO TO 20
61000 POWER=0
      GO TO 20
65000 DECMPT=.TRUE.
      GO TO 20
67000 CONTINUE
      NUMREG=0
      PT=0
      POWER=0
      DECMPT=.FALSE.
      GO TO 20
```

```
C                          SPECTRUM RECEIVER CONTROL
69000 ISTAT=KTLWR(5,Y'1C0')
       GO TO 20
70000 ISTAT=KTLWR(5,Y'180')
       GO TO 20
71000 ISTAT=KTLWR(5,Y'100')
       GO TO 20
C                          PROGRAM CONTROL
72000 CMD=NUM-71
      GO TO 20
74000 HOLD=1
      GO TO 20
9999  WRITE (4,405)
405    FORMAT('SORRY I AM NOT PROGRAMED',/'FOR THAT ONE YET',/)
       CALL ILEHDL
C
C              LIGHT A LAMP (NUM)
C
 20   CALL KTLWR(4,NUM)
  10  CALL KTLEN(4,JUNK)
      CALL KTLRET
      END
```

```
$TITL SUBROUTINE ILLEGAL ENTRY HANDLER
      SUBROUTINE ILEHDL
C
C
C
C BRUNER LIBRARY SUBROUTINE
C
C
C TODD T.W. BRUNER
C APRIL 9, 1978
C SATCOM SIGNAL ANALYZER LAB, MONTEREY, CALIFORNIA
C
C
C PURPOSE: INFORM USER THAT
C      HE PUSHED THE WRONG BUTTON
C      ENTRY IS STILL REQUIRED
C      RETURN TO CALL ING PROGRAM
C-----------------------------------------------------------------
      CALL KTLWR(4,62)
C LEAVE IT ON FOR A WHILE
      CALL WAITMS(1000)
C TURN IT OFF
      CALL KTLWR(4,-62)
C TELL HIM TO ENTRE A NUMBER
      CALL KTLWR(4,63)
      CALL KTLEN (4,JUNK)
      CALL KTLRET
      END
```

```
C       WAITMS   :   SUBROUTINE TO DELAY THE CALLER
C
C       WAITMS HAS ONE CALLING PARAMETER, THE LENGTH OF TIME,
C         IN MILLISECONDS, THAT THE CALLER'S TASK IS TO BE
C         DELAYED.
C       WAITMS USES SVC 2 TO GENERATE A TRUE TIMED DELAY.
C
      SUBROUTINE WAITMS(MS)
      INTEGER MS
C
      I=MS
$ASSM
      ST    11,PARAM
      SVC   2,BLOCK
      BS    DONE
      ALIGN 4
BLOCK DB    0,11
PARAM DCF   0
DONE  EQU   *
$FORT
      END
```

```
SUBROUTINE TEKTRO
CALL INITT(960)
CALL TERM(3,4096)
CALL SETBUF(2)
RETURN
END
```

The following program is titled SPRCVR. It is a FORTRAN program designed to control the spectrum receiver by the C4. This program exists as task and is easily executed by use of the SPRCVR CSS command. One need only type on the system console SPRCVR, after ensuring that SPRCVR.TSK and SPRCVR.CSS exist on the system volume (disc). The SPRCVR CSS command follows the SPRCVR FORTRAN program.

```
C SPECTRUM RECEIVER CONTROL PROGRAM
C THIS PROGRAM IS VERY SIMILAR TO THE SPECTRL PROGRAM (C4)
C IT HAS ONLY THE PURPOSE OF CONTROLING THE SPECTRUM
C RECEIVER (A25). THIS PROGRAM WILL CONTROL ONLY ONE
C LOCAL OSCILLATOR. THE OSC THAT IT CONTROLS IS IN RACK
C 17 A13. THIS IS THE OSCILLATOR TO SPECHANL C. ANY
C OTHER OSCILLATORS MUST BE SET BY HAND.
C THE FREQ CAN BE CHANGED BY SELECTING A FREQUENCY
C PLAN AND CHANNEL OR BY ENTERING THE FREQUENCY DESIRED.
C NOTE THAT THIS PROGRAM DIFFERS FROM C4 IN THAT
C 150 MHZ IS NOT SUBTRACTED FROM THE CENTER FREQUENCE
C ENTERED. WHAT YOU ENTER IS WHAT YOU WILL GET
C
      LOGICAL DECMPT
      INTEGER*4 PT,POWER,STACK(3),PONTR,STATE,LGTAB
     *,IDEV,NUM,NUMBER(16),IFREQ(24,4)
     *,JFREQ,MODE,ICHN,REMBER,INDEX,IIFREQ,IFATN
      INTEGER*4 LGTAB(78,2)
       ADDRESS INTRPT
      DATA PT,POWER,NUMREG/3*0/
      DATA NUMBER/7,8,9,0,0,4,5,6,0,1,2,3,4*0/
      DATA LGTAB/2*1,10*0,5*1,19*0,2*1,4*0,1,25*0,3*1,0,0,5*0
     1,48*0,17*1,0,1,11*0/
      DATA IFREQ /988500000,988750000,989000000,
     C989250000,989500000,989750000,990000000,990250000,
     C990500000,990750000,991000000,991250000,991500000,
     C991750000,992000000,992250000,992500000,992750000,
     C993000000,993250000,1041500000,1075500000,
     C0,0,1004500000,1019500000,1036500000,1053500000,
     C1069500000,1084500000,1153500000,1168500000,1182500000,
     C1197500000,939450000,939550000,939600000,939650000,
     C939700000,939750000,939800000,939850000,939900000,
     C939950000,940000000,940100000,1106000000,0,
     C1005500000,1020500000,1037500000,1054500000,1070500000,
     C1085500000,1154500000,1169500000,1183500000,1198500000,
     C940450000,940550000,940600000,940650000,940700000,
     C940750000,940800000,940850000,940900000,940950000,
     C941000000,941100000,1117000000,0,
     C1006500000,1021500000,1038500000,1055500000,
     C1071500000,1086500000,1155500000,1170500000,1184500000,
     C1199500000,941450000,941550000,941600000,941650000,
     C941700000,941750000,941800000,941850000,941900000,
     C941950000,942000000,942100000,1123000000,0/
      DO 100 I=1,73
      CALL KTLWR(4,-I)
100   CONTINUE
      STATE=1
       PONTR=1
      JFREQ=1
      CALL KTLWR(4,1)
      INTRPT=A'1111'
      CALL KTLON(4,INTRPT)
```

```
C IF YOU DO NOT GET AN INTERUPT PLEASE WAIT
1100    CALL KTLWAT
1111   CALL KTLRD(IDEV,NUM)
C               TURN OFF ENTRY REQUIRED
        CALL KTLWR(4,-63)
        NUM=NUM
       STATE=STATE
       LGTAB(NUM,STATE)=LGTAB(NUM,STATE)
       IF (LGTAB(NUM,STATE).EQ.0) CALL ILEHDL
       IF((NUM.GT.48).OR.((NUM.LT.37).AND.(NUM.GT.9))
      1.OR.(NUM.LT.8))GO TO 200
       STACK(PONTR)=STATE
       PONTR=PONTR+1
       STATE=2
200    CONTINUE
       GO TO (1000,1000,1000,1000,9999,9999,
      *7000,8000,9000,10000,11000,9999,
      *13000,14000,15000,16000,17000,9999,
      *19000,19000,19000,19000,19000,19000,
      *9999,9999,27000,28000,29000,9999,
      *31000,9999,33000,9999,9999,9999,
      *37000,38000,39000,40000,41000,9999,
      *43000,44000,45000,9999,9999,9999,
      *49000,49000,49000,52000,53000,49000,49000,49000,
      *57000,49000,49000,49000,61000,9999,9999,49000,
      *65000,9999,67000,9999,69000,70000,71000,
      *72000,72000,74000,72000,72000,9999),NUM
1000   JFREQ=NUM
       GOTO 20
7000   GO TO 9999
8000   GO TO 9999
9000   GO TO 9999
10000  GO TO 9999
11000  GO TO 9999
13000  ISTAT= KTLWR(5,Y'220')
        GO TO 20
14000  ISTAT= KTLWR(5,Y'228')
        GO TO 20
15000  ISTAT= KTLWR(5,Y'230')
          GO TO 20
16000  ISTAT=KTLWR(5,Y'238')
             GO TO 20
17000  ISTAT=KTLWR(5,Y'218')
       GO TO 20
19000  GO TO 9999
27000  GO TO 9999
28000  GO TO 9999
29000  GO TO 9999
31000  GO TO 9999
33000  GO TO 9999
37000  REMBER=NUM
       GO TO 20
```

```
38000 REMBER=NUM
      GO TO 20
39000 GO TO 9999
40000 GO TO 9999
41000 GO TO 9999
42000 GO TO 9999
43000 REMBER=NUM
      GO TO 20
44000 GO TO 9999
45000 GO TO 9999
49000 CONTINUE
      IF (DECMPT) PT=PT+1
C     NOW DO THE TRANSLATION
      NUMREG=NUMREG*10+NUMBER(NUM-48)
      B=NUMREG*10.0**(POWER-PT)
      GOTO 20
52000 POWER=6
      GO TO 20
53000 CONTINUE
50    A=NUMREG
       W=FLOAT(POWER-PT)
       FNUMRG=A*10.0**(W)
       NUMREG=FNUMRG
      PT=0
      POWER=0
      INDEX=REMBER-36
      PONTR=PONTR-1
      STATE=STACK(PONTR)
      GO TO (3700,3800,3900,4000,4100,9999,4300,4400,4500,4600,4700
     *),INDEX
3700  ICHN=NUMREG
      NUMREG=0
      PT=0
      POWER=0
      MODE=10
      CALL KTLWR(7,IFREQ(ICHN,JFREQ))
      GO TO 20
3900  GO TO 9999
4000  GO TO 9999
4100  GO TO 9999
4400  GO TO 9999
4500  GO TO 9999
4600  GO TO 9999
4700  GO TO 9999
3800  C=FNUMRG
      IIFREQ=C*10
      NUMREG=0
      PT=0
      POWER=0
      CALL KTLWR(7,IIFREQ)
      GO TO 20
4300  IFATN=NUMREG
```

```
      NUMREG=0
      PT=0
      POWER=0
      IHEX=X'37F'-IFATN
      ISTAT=KTLWR(5,Y'37F'-IFATN)
      GO TO 20
57000 POWER=3
      GO TO 20
61000 POWER=0
      GO TO 20
65000 DECMPT=.TRUE.
      GO TO 20
67000 CONTINUE
      NUMREG=0
      PT=0
      POWER=0
      DECMPT=.FALSE.
      GO TO 20
69000 ISTAT=KTLWR(5,Y'1C0')
       GO TO 20
70000 ISTAT=KTLWR(5,Y'130')
       GO TO 20
71000 ISTAT=KTLWR(5,Y'100')
      GO TO 20
72000 CONTINUE
74000 CONTINUE
9999   CALL ILEHDL
 20   CALL KTLWR(4,NUM)
  10  CALL KTLEN(4,JUNK)
      CALL KTLRET
      END
```

```
L SPRCVR;T SPRCVR;OPT NON;ST
$EXIT
```

The following program is a CSS command. It is titled BIGBOY. Its purpose is to permit six FORTRAN programs to be compiled by the operator. This command is particularly useful when one desires to make subroutines whose object module is to be included in the TET file; or is one is making a library of subroutines. There are six passing arguments (@1,@2,@3,...@6). The arguments are the names of the programs to be operated on.

```
$COPY
*I AM FORTOBJING @1,@2,@3,@4,@5,@6
$NOC
SET SYS 15
$JOB
FORTOBJ @1
$IFNE 0;$SKIP
$ENDC
$TERMJOB
$COPY
*I AM FORTOBJING @2
$NOC
$JOB
FORTOBJ @2
$IFNE 0;$SKIP
$ENDC
$TERMJOB
$COPY
*I AM FORTOBJING @3
$NOC
$JOB
FORTOBJ @3;$IFNULL@3;$SKIP
$IFNE 0;$SKIP
$ENDC
$TERMJOB
$COPY
*I AM FORTOBJING @4
$NOC
$JOB
FORTOBJ @4;$IFNULL@4;$SKIP
$IFNE 0;$SKIP
$ENDC
$TERMJOB
$COPY
*I AM FORTOBJING @5
$NOC
$JOB
FORTOBJ @5;$IFNULL@5;$SKIP
$IFNE 0;$SKIP
$ENDC
$TERMJOB
$COPY
*I AM FORTOBJING @6
$NOC
$JOB
FORTOBJ @6;$IFNULL@6;$SKIP
$IFNE 0;$SKIP
$ENDC
$TERMJOB
$COPY
*>>>>>>>>>>>>>>>>>>>>>>>>>BIGBOY IS DONE<<<<<<<<<<<<<<<<<<<<<<
```

```
$NOC
SET SYS 10
$EXIT
```

BIGPR is the second CSS command which follows. It is program similar to BIGBOY. Instead of operating on the programs, the command will print the programs on the printer. BIGPRTH is not shown, but it also exists. This program will print the desired programs on the printer in the format necessary for a thesis.

```
$JOB
PR @1.FTN;$IFNULL@1.FTN;$SKIP
$TERMJOB
$JOB
PR @2.FTN;$IFNULL@2.FTN;$SKIP
$TERMJOB
$JOB
PR @3.FTN;$IFNULL@3.FTN;$SKIP
$TERMJOB
$JOB
PR @4.FTN;$IFNULL@4.FTN;$SKIP
$TERMJOB
$JOB
PR @5.FTN;$IFNULL@5.FTN;$SKIP
$TERMJOB
$JOB
PR @6.FTN;$IFNULL@6.FTN;$SKIP
$TERMJOB
$COPY
* I HAVE PRINTED @1,@2,@3,@4,@5,@6
* ANYTHING MORE ??????
$NOC
$EXIT
```

TAPIT, another CSS command follows next. Similar to BIGBOY, BIGPR, and BIGPRTH, TAPIT will copy six programs to the left cassette drive. It should be clear with these examples of BIGBOY, BIGPR, BIGPRT4, and TAPIT, that the user can link a number of operator commands together. To master this early will save the user time.

```
$JOB
COPYA @1,CAS1:
$TERMJOB
$JOB
COPYA @2,CAS1:
$TERMJOB
$JOB
COPYA @3,CAS1:
$TERMJOB
$JOB
COPYA @4,CAS1:
$TERMJOB
$JOB
COPYA @5,CAS1:
$TERMJOB
$JOB
COPYA @6,CAS1:
$NOC
$EXIT
```

The next program is B, another CSS command. This command will cause the left cassette file to be copied into a file called NEW. The file NEW is then displayed on the screen. The purpose of this command is to permit one to easily examine files on a cassette.

```
COPYA CAS1:,NEW
SH NEW
EXIT
```

The AP CSS command follows. This program allocates all the necessary files and then will start the Floating Point System programs for software development.

```
$IFNX @1.APO;ALL @1.APO,IN,80
$ENDC
$IFNX @1.APE;ALL @1.APE,IN,80
$ENDC
$IFNX @1.CAL;ALL @1.CAL,IN,80
$ENDC
L .BG,MON1:APAL;T .BG;AS 4,CON:
AS 5,CON:;AS 6,CON:;ST
$IFE 0
L .BG,MON1:APLINK;T .BG;AS 4,CON:
AS 5,CON:;AS 6,CON:;ST
$IFE 0
L .BG,MON1:APSIM;T .BG;AS 4,CON:
AS 5,CON:;AS 6,CON:;ST
$ENDC
```

BACKUP is the last CSS command. This will backup a disc that has been placed in the third disc slot. The comments in the listing describe the program.

```
$COPY
*THIS IS A PROGRAM TO BACKUP A DISC.
*IT WILL FIRST CLEAR D4(THE BOTTOM DISC).
*A COMPRESS IS DONE FROM D3 TO D4
*A DISCINIT IS DONE ON D3
*A COMPRESS IS DONE FROM D4 TO D3
*A COMPRESS IS DONE FROM D4 TO THE MAG TAPE
*THE DISC TO BE BACKED UP MUST BE IN D3
*THE ONLY PASSING ARGUMENT IS THE NAME OF THE
*DISC TO BE BACKED UP(OR NEW NAME).
*INSURE THAT THERE IS A TAPE ON THE DRIVE
$NOC
L .BG,DISCINIT,48;T .BG;OPT NON;ST ,D=D4:,V=COMP,C,B=48
$IFE 0
L .BG,C3COMP,48;T .BG;MA D4:,ON;ST ,I=D3:,O=D4:,V
$IFE 0
V COMP;V COMP/TE
MA D3:,OF
L .BG,DISCINIT,48;T .BG;ST ,D=D3:,V=@1,C,B=48
$IFE 0
L .BG,C3COMP,48;T .BG;MA D3:,ON;ST ,I=D4:,O=D3:,L=CON:
$IFE 0
L .BG,C3COMP,48;T .BG;ST ,I=D4:,O=MT:,L=CON:
$IFE 0
REW MT:
D D;D M;V
$NOC
$ENDC
$EXIT
$CLEAR
```

APPENDIX V

The programs in this appendix are still in development or are complete modules that are to be integrated with pending modules into one program which is to replace the program SATCOM. The order and use follow:

ORDER	NAME	USE
1st	IOPACK	Test program for the I/O package. This is a complete module and is waiting integration with other modules.
2nd	AXIS	Draws axis on the Tektronix.
3rd	XGRATL	Draws graticle lines in y direction.
4th	YGRATL	Draws graticle lines in x direction.
5th	GRATCL	Labels the graticles in the x or y plane.
6th	SCREEN	Draws and labels the graticles in the xy plane.
7th	ANNOT	Draws axis and graticles in xy plane.
8th	VIRWIN	Sets the vertual window for SCREEN.
9th	RECTRL (SPRCVR)	Controls receiver for FULL MON button in SPECTRL program. This is a complete working module.
10th	SPACQ	Controls the DAU for a-1 acquisitions. This is a complete module. It incorporates the pre-start circuit conditions. All data returned in this program is good data. There are no previous words of data that must be ignored.
11th	PRCDTA	Indevelopment program to process the data for frequency and time domain analysis.
12th	FFFT	Indevelopment program to execute the FFT algorithm.
13th	AVG	Program to average data. This program is a complete module.
14th	SCAIL	Indevelopment program to scale data for high speed to integer plotting on the Tektronix.
15th	GETIT	Program to get data from the AP-12ØB. This program is a complete module.
16th	GRAFIT	Temporary debugger program for development of the FUL MON program.

```
C       MAX AND MIN REFER TO TH E VALUES IN USER UNITS ,I.E.
C            APPLES,FREQ,DB,ETC.
C            OF THE ITEM THAT IS TO BE PLOTTED
C       VIRWIN COMPUTES THE SPACING BASEC ON THE NUMBER OF
C            TICS YOU WANT(NTIC)
C            AND LEAVES A HALF SPACING BOARDER
C       OFSET INDICATES THE LOCATION ON THE OTHER AXIS YOU'D
C            LIKE AN AXIS TO
C              BE DRAWN.
      INTEGER SCXSTR,SCXEND,SCYSTR,SCYEND
      DATA SCXSTR,SCXEND,SCYSTR,SCYEND/0200,4000,0200,3000/
33    WRITE(5,10)
10     FORMAT('INPUT XMIN,XMAX,NXTIC ONE LINE AT ATIME F12.5,I2')
      READ(5,11)XMIN,XMAX,NXTIC
11     FORMAT(2(F12.5/),I2)
      WRITE(5,20)
20     FORMAT('INPUT YMIN,YMAX,NYTIC ONE LINE AT ATIME F12.5,I2')
      READ(5,11)YMIN,YMAX,NYTIC
      WRITE(5,30)
30     FORMAT('INPUT XOFSET YOFSET  2F12.5')
      READ(5,31)XOFSET,YOFSET
31     FORMAT (2F12.5)
      WRITE(5,50)
50    FORMAT(' INPUT THE CHARACTER SIZE 1,2,3,4 ')
      READ(5,51)ICHAR
51    FORMAT (I1)
      CALL TEKTRO
      CALL VIRWIN(XMAX,XMIN,YMAX,YMIN,NXTIC,NYTIC)
      CALL ANNOT(XMIN,XMAX,NXTIC,YOFSET,
     *YMIN,YMAX,NYTIC,XOFSET,
     *SCXSTR,SCXEND,SCYSTR,SCYEND,ICHAR)
      CALL TSEND
      GO TO 33
      END
```

```
      SUBROUTINE AXIS(XMIN,XMAX,YOFSET,YMIN,YMAX
     *,XOFSET)
100   CALL MOVEA(XMIN,YOFSET)
200   CALL DRAWA(XMAX,YOFSET)
300   CALL MOVEA(XOFSET,YMIN)
      CALL DRAWA(XOFSET,YMAX)
       XOFSET=XOFSET
       YOFSET=YOFSET
       XMAX=XMAX
       YMAX=YMAX
       XMIN=XMIN
       YMIN=YMIN
      RETURN
      END
```

```
      SUBROUTINE XGRATL(XLBL,YMIN,YMAX)
      CALL MOVEA(XLBL,YMIN)
      CALL DASHA(XLBL,YMAX,1)
      CALL MOVEA(XLBL,YMIN)
      RETURN
      END
```

```
SUBROUTINE YGRATL(YLBL,XMIN,XMAX)
CALL MOVEA(XMIN,YLBL)
CALL DASHA(XMAX,YLBL,1)
CALL MOVEA(XMIN,YLBL)
RETURN
END
```

```
      SUBROUTINE GRATCL(II,MAX,MIN,NTIC,OMIN,OMAX,ICHAR)
      INTEGER BLK(33)
      REAL MAX,MIN,LBL
      SPACE=(MAX-MIN)/(NTIC-1.0)
      CALL CHRSIZ(ICHAR)
      DO 10 J=1,NTIC
      LBL=MIN+SPACE*(J-1)
      ENCODE(BLK,200)LBL
200   FORMAT(E10.3)
      IF(ABS(LBL-MIN).LE.(SPACE/2))GO TO 10
      IF (II.EQ.1)GO TO 1
      CALL YGRATL(LBL,OMIN,OMAX)
      GO TO 2
1     CALL XGRATL(LBL,OMIN,OMAX)
2     CALL ANBSTR(BLK,10)
10    CONTINUE
      RETURN
      END
```

```
 SUBROUTINE SCREEN(XMAX,XMIN,NXTIC,YMAX,YMIN,NYTIC,
*ICHR)
 II=1
 CALL XGRATL(XMIN,YMIN,YMAX)
 CALL GRATCL(II,XMAX,XMIN,NXTIC,YMIN,YMAX,ICHAR)
 II=0
 CALL YGRATL(YMIN,XMIN,XMAX)
 CALL GRATCL(II,YMAX,YMIN,NYTIC,XMIN,XMAX,ICHAR)
 RETURN
 END
```

```
$TRCE
      SUBROUTINE ANNOT(XMIN,XMAX,NXTIC,YOFSET,
     *YMIN,YMAX,NYTIC,XOFSET,
     *SCXSTR,SCXEND,SCYSTR,SCYEND,ICHAR)
      INTEGER SCXSTR,SCXEND,SCYSTR,SCYEND
10    CALL TWINDO(SCXSTR,SCXEND,SCYSTR,SCYEND)
20     CALL AXIS(XMIN,XMAX,YOFSET,YMIN,YMAX,XOFSET)
30    CALL SCREEN(XMAX,XMIN,NXTIC,YMAX,YMIN,NYTIC,ICHAR)
      RETURN
      END
```

```
 SUBROUTINE VIRWIN(XMAX,XMIN,YMAX,YMIN,NXTIC,
*NYTIC)
 XSPACE=(XMAX-XMIN)/(NXTIC-1.0)
 XMINT=XMIN-XSPACE/2.0
 XMAXT=XMAX+XSPACE/2.0
 YSPACE=(YMAX-YMIN)/(NYTIC-1.0)
 YMINT=YMIN-YSPACE/2.0
 YMAXT=YMAX+YSPACE/2.0
 CALL DWINDO(XMINT,XMAXT,YMINT,YMAXT)
 RETURN
 END
```

```
      SUBROUTINE SPRCVR(SFREQC,SFREQB,SPMDE)
      INTEGER SPCMD(3,3),SPMDE,SFREQC,SFREQB
      DATA SPCMD/Y'37F',Y'230',Y'100',Y'100'
     *Y'37F',Y'218',3*0/
      CALL KTLWR(7,SFREQC)
      CALL KTLWR(9,SFREQB)
      DO 5 I=1,3
5     CALL KTLWR(5,SPCMD(I,SPMDE))
      RETURN
      END
```

```
      SUBROUTINE SPACQ(FREQ,DAUMDE,DAUDTA,BLKSIZ)
      INTEGER*2 DAUCMD(7,2),DAUDTA(2050),ICODE
      INTEGER DAUMDE,BLKSIZ
      DATA DAUCMD/0,4,0,3,16,5,17,0,4,0,3,32,5,34/
C                                          SET SAMPLE RATE
      CALL BCDFRQ(FREQ)
      CALL WAITMS(0150)
      DAUCMD(3,DAUMDE)=BLKSIZ
      DO 5 I=1,6,2
5     CALL DAUREG(DAUCMD(I,DAUMDE),DAUCMD(I+1,DAUMDE))
      CALL DAURUN(DAUCMD(7,DAUMDE),ICODE)
C                                          GET THE DATA
      DAUCMD(3,DAUMDE)=2*DAUCMD(3,DAUMDE)
      DAUCMD(1,DAUMDE)=1
      CALL DAUGET(DAUDTA,DAUCMD(1,DAUMDE),DAUCMD(3,DAUMDE))
      RETURN
      END
```

```
      SUBROUTINE PRCDTA(DAUDTA,BLKSIZ,TIMDM,RESULT,I,BLKAVG)
      INTEGER*2 DAUDTA
      INTEGER BLKSIZ,TIMDM,RESULT,SOURCE(2),BLKAVG
$TRCE
      SOURCE(1)=0
      SOURCE(2)=2048
      J=1
      IF (MOD(I,2).EQ.0)J=2
5     CALL APPUT(DAUDTA,SOURCE(J),2*BLKSIZ,1)
       SOURCE(J)=SOURCE(J)
C PUT THE APWD AT STATEMENT 20 AFTER THE FREQ DOMAIN WORKS
      CALL APWD
      IF(TIMDM.EQ.1)GO TO 20
      III=3-J
10    CALL FFFT(SOURCE(III),BLKSIZ,I)
      SOURCE(III)=SOURCE(III)
      RESULT=6144
      RETURN
20    CALL VFLT(SOURCE(1),1,SOURCE(1),1,BLKSIZ*2)
      RESULT=0
100   RETURN
      END
```

```
      SUBROUTINE FFFT(SOURCE,BLKSIZ,J)
      INTEGER SOURCE,BLKSIZ,DESTIN,SUM
      REAL STOR(2050)
      DATA DESTIN,SUM/4096,6144/
      CALL VFLT(SOURCE,1,SOURCE,1,2*BLKSIZ)
      CALL APWR
      CALL APGET(STOR,SOURCE,BLKSIZ,2)
      CALL APWD
      WRITE(8,13)(STOR(K),K=1,BLKSIZ)
      CALL CFFTB(SOURCE,4096,BLKSIZ,1)
      CALL APWR
      CALL APGET(STOR,4096,BLKSIZ,2)
      CALL APWD
      WRITE(8,13)(STOR(K),K=1,BLKSIZ)
13    FORMAT(8F14.5)
30000 IF(J.NE.1)CALL SCJMA(4096,1,6144,1,6144,1,
     *BLKSIZ)
      IF(J.EQ.1)GO TO 40000
      CALL APWR
      CALL APGET(STOR,6144,BLKSIZ,2)
      CALL APWD
      WRITE(8,13)(STOR(K),K=1,BLKSIZ)
40000 IF(J.EQ.1)CALL VCLR(6144,1,BLKSIZ)
      RETURN
      END
```

```
      SUBROUTINE AVG(BLKSIZ,BLKAVG,SOURCE)
      INTEGER BLKSIZ,BLKAVG,SOURCE
      FACTOR=1.0/(BLKSIZ*BLKAVG)
      IPUT=15380
      CALL APPUT(FACTOR,IPUT,1,2)
C                  REMOVE THIS WAIT CMD AND SEE WHAT HAPPENS
C                  IT MIGHT WORK WITHOUT IT BECAUSE ONLY
C                  ONE NUMBER IS GOING IN AND MIGHT NOT
C                  NEED TO WAIT SO LONG
      CALL APWD
      CALL VSMUL(SOURCE,1,IPUT,SOURCE,1,BLKSIZ)
      RETURN
      END
```

```
      SUBROUTINE SCAIL(SOURCE,BLKSIZ,IYPT,SCALE,TIMDM)
      INTEGER SOURCE,BLKSIZ,IYPT,TIMDM,VVMAX,VVMIN
     *,RANGE,YPTRNG,SCAL
      INTEGER*2 IDIOT
      REAL SCALE(20),STOR(2050)
      DATA VVMAX,VVMIN,RANGE,SCAL,YPTRNG/8360,8364,8368,/
     *8372,8376
$TRCE
      SOURCE=SOURCE
      BLKSIZ=BLKSIZ
      IDIOT=IYPT
      CALL VCLR(VVMAX,1,20)
      CALL APPUT(IDIOT,YPTRNG,1,1)
      CALL APPUT(IDIOT,YPTRNG+1,1,1)
      CALL APWD
      J=1
      IF(TIMDM.EQ.1)J=2
      DO 43 I=1,J
300   CALL MAXV(SOURCE+(I-1),J,VVMAX+(I-1),BLKSIZ)
400   CALL MINV(SOURCE+(I-1),J,VVMIN+(I-1),BLKSIZ)
43     CONTINUE
10    CALL VSUB(VVMIN,1,VVMAX,1,RANGE,1,2)
600   CALL VFLT(YPTRNG,1,YPTRNG,1,1)
700   CALL VDIV(RANGE,1,YPTRNG,1,SCAL,1,2)
      CALL APWR
      CALL APGET(SCALE,VVMAX,20,2)
      CALL APWD
      DO 63 I=1,20
      SCALE(I)=SCALE(I)
63    CONTINUE
      DO 53 I=1,J
      SOURCE=SOURCE
      SCAL=SCAL
      J=J
      BLKSIZ=BLKSIZ
         IF(TIMDM.EQ.1)BLKSIZ=2*BLKSIZ
      CALL APWR
      CALL APGET(STOR,SOURCE+(I-1),BLKSIZ,2)
      CALL APWD
      DO 737 KK=1,BLKSIZ
      STOR(KK)=STOR(KK)
737    CONTINUE
         IF(TIMDM.EQ.1)BLKSIZ=BLKSIZ/2
800   CALL VSMUL(SOURCE+(I-1),J,SCAL+(I-1),SOURCE+(I-1),J,BLKSIZ)
      CALL APWR
      CALL APGET(STOR,SOURCE+(I-1),BLKSIZ,2)
      CALL APWD
      DO 73 KK=1,BLKSIZ
      STOR(KK)=STOR(KK)
73    CONTINUE
53    CONTINUE
20    RETURN
```

```
      SUBROUTINE GETIT(PRODTA,SOURCE,BLKSIZ)
      INTEGER SOURCE,BLKSIZ
      INTEGER*2 PRODTA(2050)
550   CALL VFIX(SOURCE,1,SOURCE,1,BLKSIZ)
560   CALL APWR
570   CALL APGET(PRODTA,SOURCE,BLKSIZ,1)
      CALL APWD
      RETURN
      END
```

```
      SUBROUTINE GRAFIT(IXST,IYST,IXMXPT,NPTS,
     *IDATA,TIMDM)
      INTEGER*2 IDATA(2050)
      INTEGER TIMDM
      INCRMT=IXMXPT/NPTS
      J=1
      IF(TIMDM.EQ.1)J=2
450   CALL MOVABS(IXST,IYST+IDATA(1))
      DO 5 I=1,NPTS,J
      CALL DRWABS(IXST+INCRMT*(I),IDATA(I+1)+IYST)
5     CONTINUE
      CALL MOVABS(0000,3120)
460   CALL TSEND
      RETURN
      END
```

LIST OF ABBREVIATIONS

AC	- Alternating Current
ADC	- Analog-to-Digital Conversion
CSS	- Command Substitution System
DAU	- Data Acquisition Unit
FFT	- Fast Fourier Transform
FLTSAT	- Fleet Satellite
FSM	- Fleet Satellite Communications Spectrum Monitor
HP	- Hewlett-Packard
IEEE	- Institute of Electronic and Electrical Engineers
I/O	- Input-Output
NAVELEX	- Naval Electronic Systems Command
NAVPGSCOL	- Naval Postgraduate School
NPS	- Naval Postgraduate School
PME	- Project Management Electronics
RF	- Radio Frequency
RFI	- Radio Frequency Interference
RS	- Recommended Standard
SATCOM	- Satellite Communications (also appears as a name for a program in Chapter 3)
SPCHNL A,B,C	- Spectrum Receiver Channel A or B or C
SPECTRL	- Spectrum Control Program
SSA	- Satellite Signal Analyzer
SVC	- Supervisor Calls
UHF	- Ultra High Frequency

LIST OF REFERENCES

1. Naval Postgraduate School Report 52AB76031, Evaluation of the AS 3018/WSC-1(V) Shipboard SATCOM Antenna, by R. W. Adler, B. K. Hollar, and J. E. Ohlson, Confidential, March 1976.

2. Naval Postgraduate School Report62OL76105, Deck Boxes for UHF SATCOM Radio Frequency Interference Study, by G. B. Parker and J. E. Ohlson, October 1976.

3. Naval Postgraduate School Report 62OL76106, A High Level Noise Blanker and RF Amplifier System for the UHF Band, by F. E. Mace and J. E. Ohlson, October 1976.

4. Naval Postgraduate School Report 62OL76107, A Level Density Analyzer for Shipboard RFI Measurements, by D. C. Arneson and J. E. Ohlson, October 1976.

5. Naval Postgraduate School Report 62OL76108, Instrumentation Package for Measurement of Shipboard RFI, by A. R. Shuff and J. E. Ohlson, October 1976.

6. Naval Postgraduate School Report 62OL76103, Shipboard Radio Frequency Interference in UHF Satellite Communications, by J. E. Ohlson and T. C. Landry, Confidential, October 1976.

7. Naval Postgraduate School Report 62OL76109, A Simulator for Shipboard Radio Frequency Interference in Satellite Communications, by E. S. Brick and J. E. Ohlson, October 1976.

8. Naval Postgraduate School Report 62OL76091, Receiver Desensitization of the AN/WSC-3 Satellite Communications Set, by R. F. Carlson and J. E. Ohlson, Confidential, September 1976.

9. Naval Postgraduate School Report 62OL76101, Receiver Desensitization of the AN/SSR-1 Satellite Communication Receiver, by R. F. Carlson and J. E. Ohlson, Confidential, October 1976.

10. Naval Postgraduate School Report 62OL76102, Bit Error Rate Measurements on the AN/WSC-3 and AN/SSR-1 Satellite Communications Sets, by R. F. Carlson and J. E. Ohlson, October 1976.

11. Naval Postgraduate School Technical Report NPS62OL78002, Receiver Design for the Naval Postgraduate School SATCOM Signal Analyzer, by C. B. Williams and J. E. Ohlson, January 1978.

12. Naval Postgraduate School Technical Report NPS6278001, Digital Control and Processing for a Satellite Communication Monitoring System, by G. W. Bohannan and J. E. Ohlson, January 1978.

13. Naval Postgraduate School Technical Report NPS62OL78003, Hardware Development for a Satellite Signal Analyzer (U), by J. D. Zuber, Jr. and J. E. Ohlson, December 1977.

14. Naval Postgraduate School Technical Report, System Development for Satellite Oscillator Stability Measurements, by Bobbie G. Edgington and J. E. Ohlson, June, 1978.

15. Naval Postgraduate School Technical Report, Digital Control and Interfacing for a High Speed Satellite Communications Signal Processor, by Richard Randolph Mead and J. E. Ohlson, September 1978.

16. Naval Postgraduate School Technical Report, Data Acquisition Unit for SATCOM Signal Analyzer, by Marvin J. Langston and J. E. Ohlson, June 1978.

17. Meyers, G. J., Software Reliability, Principles and Practices, Wiley & Sons, April 1976.

18. Naval Postgraduate School Technical Report, Implementation of Control Bus Hardware and Utilization of Satellite Signal Analyzer, by Ronald M. Thomas and J. E. Ohlson, September 1978.

19. Interdata, Inc., OS/32 Operators Reference Manual, Publication Number 29-574, September 1977.

20. Floating Point Systems, Inc., Processor Handbook, Publication Number 7259-02, Rev 2, May 1976.

21. Floating Point Systems, Inc., Software Development Package Manuals, Publication Number 7292, February 1976.

22. Floating Point Systems, Inc., Programmer's Reference Manual, Publication Number 7319, January 1978.

23. Floating Point Systems, Inc., AP-12ØB Math Library, Rev 2, Publication Number 7288-03, August 1977.

24. Floating Point Systems, Inc., Vector Function Chainer Manual, Publication Number 7351, October 1977.

25. Hewlett-Packard Co., HP9830A Calculator Operating and Programming Manual, Publication Number 09830-90001, September 1973.

26. Hewlett-Packard Co., HP9830A Calculator 11274B String Variables ROM Operating Manual, Publication Number 09830-90002, January 1975.

27. Hewlett-Packard Co., 9830A Calculator Extended I/O ROM Operating Manual for 11272B & Option 272, Publication Number 09830-90007, November 1973.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Commander (Attn: E. L. Warden, PME-106-112A) Naval Electronic Systems Command Department of the Navy Washington, D.C. 20360	8
2.	Commander (Attn: W. C. Willis, PME-106-11) Naval Electronic Systems Command Department of the Navy Washington, D.C. 20360	1
3.	Commander (Attn: W. R. Coffman, PME-106-16) Naval Electronic Systems Command Department of the Navy Washington, D.C. 20360	1
4.	Library Naval Postgraduate School Monterey, California 93940	2
5.	Office of Research Administration (012A) Naval Postgraduate School Monterey, California 93940	1
6.	Professor John E. Ohlson Code 620L Naval Postgraduate School Monterey, California 93940	20
7.	Commander (Attn: LT Gary W. Bohannon, G60) Naval Security Group 3810 Nebraska Avenue, N.W. Washington, D.C. 20390	1
8.	Commander (Attn: Robert S. Trible, 0252) Naval Electronic Systems Engineering Activity (NESEA) Patuxent River, Maryland 20670	1